



Amazon EMR Serverless 使用者指南

Amazon EMR



Amazon EMR: Amazon EMR Serverless 使用者指南

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

Table of Contents

什麼是 Amazon EMR Serverless ?	1
概念	1
發行版本	1
應用程式	1
作業執行	2
工作程序	2
預先初始化的容量	3
EMR Studio	3
入門的先決條件	4
註冊 AWS 帳戶	4
授予許可	4
授與程式設計存取權	6
設定 AWS CLI	7
開啟 主控台	8
開始使用	9
許可	9
儲存	9
互動式工作負載	9
建立任務執行期角色	10
從 主控台入門	15
步驟 1：建立 應用程式	16
步驟 2：提交任務執行或互動式工作負載	16
步驟 3：檢視應用程式 UI 和日誌	19
步驟 4：清理	19
從 入門 AWS CLI	20
步驟 1：建立 應用程式	20
步驟 2：提交任務執行	21
步驟 3：檢閱輸出	23
步驟 4：清理	24
與 EMR Serverless 應用程式互動並設定	26
應用程式狀態	26
使用 EMR Studio 主控台	27
建立應用程式	27
從 EMR Studio 主控台列出應用程式	28

從 EMR Studio 主控台管理應用程式	28
使用 AWS CLI	29
設定應用程式	30
應用程式行為	30
在 EMR Serverless 中使用應用程式的預先初始化容量	32
預設應用程式組態	35
自訂映像	40
先決條件	31
步驟 1：從 EMR Serverless 基礎映像建立自訂映像	42
步驟 2：在本機驗證映像	42
步驟 3：將映像上傳至您的 Amazon ECR 儲存庫	43
步驟 4：使用自訂映像建立或更新應用程式	44
步驟 5：允許 EMR Serverless 存取自訂映像儲存庫	45
考量和限制	46
設定 EMR Serverless 應用程式的 VPC 存取以連線至資料	46
建立應用程式	47
設定應用程式	49
子網路規劃的最佳實務	50
架構選項	51
使用 x86_64 架構	51
使用 arm64 架構 (Graviton)	52
使用 Graviton 啟動新應用程式	52
將現有應用程式轉換為 Graviton	52
考量事項	53
任務並行和佇列	53
並行和佇列的主要優點	54
並行和佇列入門	54
並行和佇列的考量事項	55
上傳資料	56
先決條件	56
開始使用 S3 Express One Zone	57
執行任務	59
作業執行狀態	59
具有寬限期的任務執行取消	60
批次任務的寬限期	61
串流任務的寬限期	62

考量事項	46
使用 EMR Studio 主控台	65
提交工作	65
存取任務執行	67
使用 AWS CLI	67
執行 IAM 政策	69
開始使用	69
CLI 命令範例	69
重要說明	71
政策交集	71
使用隨機最佳化磁碟	74
主要優點	74
開始使用	75
使用無伺服器儲存	78
主要優點	79
開始使用	79
考量和限制	81
支援的 AWS 區域	81
處理持續串流資料的串流任務	82
考量和限制	83
開始使用	84
串流連接器	84
日誌管理	87
執行 EMR Serverless 任務時使用 Spark 組態	87
Spark 參數	87
Spark 屬性	90
資源組態最佳實務	95
Spark 範例	96
執行 EMR Serverless 任務時使用 Hive 組態	96
Hive 參數	97
Hive 屬性	99
Hive 範例	108
作業彈性	109
使用重試政策監控作業	112
使用重試政策記錄	112
EMR Serverless 的中繼存放區組態	113

使用 AWS Glue Data Catalog 做為中繼存放區	113
使用外部 Hive 中繼存放區	118
在 EMR Serverless 上使用 AWS Glue 多目錄階層	122
使用外部中繼存放區時的考量事項	123
跨帳戶 S3 存取	124
先決條件	124
使用 S3 儲存貯體政策	124
使用擔任的角色	125
擔任的角色範例	128
故障診斷錯誤	132
錯誤：任務失敗，因為帳戶已達到可同時使用的最大 vCPU 服務限制。	133
錯誤：任務失敗，因為應用程式已超過maximumCapacity設定。	133
錯誤：由於無法配置工作者而導致任務失敗，因為應用程式已超過maximumCapacity。	133
錯誤：S3 存取遭拒。請檢查所需 S3 資源上任務執行時間角色的 S3 存取許可。	133
錯誤：ModuleNotFoundError：沒有名為 <module> 的模組。有關如何搭配 EMR Serverless 使用 python 程式庫的使用者指南。	133
錯誤：無法擔任執行角色 <role name>，因為它不存在或未設定所需的信任關係。	133
任務層級成本分配	134
預設行為	134
如何啟用或停用此功能	134
考量事項與限制	135
執行互動式工作階段	136
所需的許可	136
使用互動式工作階段	138
考量和限制	141
執行互動式工作負載	143
概觀	143
先決條件	143
許可	143
Configuration	145
考量事項	145
透過 Apache Livy 端點執行互動式工作負載	146
先決條件	146
所需的許可	146
開始使用	148
考量事項	154

日誌記錄和監控	157
儲存日誌	157
受管儲存	158
Amazon S3	159
Amazon CloudWatch	160
輪換日誌	163
加密日誌	164
受管儲存	164
Amazon S3 儲存貯體	165
Amazon CloudWatch	165
所需的許可	165
設定 Log4j2	170
Log4j2 和 Spark	170
監控	174
應用程式和任務	174
Spark 引擎指標	182
用量指標	186
使用 EventBridge 自動化	187
EMR Serverless EventBridge 事件範例	187
標記資源	192
什麼是標籤？	192
標記資源	192
標記限制	193
使用標籤	193
教學	195
使用 Java 17	195
JAVA_HOME	195
spark-defaults	196
使用 Hudi	197
使用 Iceberg	198
使用 Python 程式庫	199
使用原生 Python 功能	199
建置 Python 虛擬環境	199
設定 PySpark 任務以使用 Python 程式庫	201
使用不同的 Python 版本	201
使用 Delta Lake OSS	203

Amazon EMR 6.9.0 版及更新版本	203
Amazon EMR 6.8.0 版及更低版本	205
從 Airflow 提交任務	205
使用 Hive 使用者定義的函數	208
使用自訂映像	209
使用自訂 Python 版本	210
使用自訂 Java 版本	210
建置資料科學映像	211
使用 Apache Sedona 處理地理空間資料	211
使用自訂映像的授權資訊	212
使用 Amazon Redshift 上的 Spark	212
啟動 Spark 應用程式	213
向 Amazon Redshift 進行身分驗證	214
讀取和寫入 Amazon Redshift	216
考量事項	218
連線至 DynamoDB	219
步驟 1：上傳至 Amazon S3	219
步驟 2：建立 Hive 資料表	220
步驟 3：複製到 DynamoDB	221
步驟 4：從 DynamoDB 查詢	222
設定跨帳戶存取權	224
考量事項	226
安全	228
安全最佳實務	229
套用最低權限準則	229
隔離不受信任的應用程式碼	229
角色型存取控制 (RBAC) 許可	229
資料保護	229
靜態加密	230
傳輸中加密	232
Identity and Access Management (IAM)	233
目標對象	233
使用身分驗證	233
使用政策管理存取權	235
EMR Serverless 如何與 IAM 搭配使用	236
使用服務連結角色	240

Amazon EMR Serverless 的任務執行期角色	245
使用者存取政策	248
標籤型存取控制的策略	252
身分型政策	255
政策更新	258
疑難排解	258
受信任的身分傳播	260
概觀	260
功能和優勢	261
運作方式	261
Trusted-Identity 傳播入門	262
互動式工作負載的受信任身分傳播	265
使用者背景工作階段	265
EMR Serverless Trusted-Identity-Propagation 整合的考量事項	269
搭配 EMR Serverless 使用 Lake Formation	270
功能可用性	271
EMR Serverless 的 Lake Formation 完整資料表存取	271
適用於 FGAC 的 Lake Formation	286
工作者間加密	312
在 EMR Serverless 上啟用交互 TLS 加密	312
使用 KMS CMK 進行磁碟加密	313
使用加密內容	313
使用客戶受管金鑰設定磁碟加密	314
磁碟加密所需的許可	316
監控金鑰用量	318
進一步了解	321
資料保護的 Secrets Manager	321
秘密的運作方式	321
建立秘密	322
指定秘密參考	322
授予對秘密的存取權	324
輪換秘密	326
資料存取控制的 S3 存取授權	326
概觀	326
啟動應用程式	327
考量事項	328

用於記錄的 CloudTrail	328
CloudTrail 中的 EMR Serverless 資訊	328
了解 EMR Serverless 日誌檔案項目	329
法規遵循驗證	331
恢復能力	331
基礎設施安全性	332
組態與漏洞分析	332
端點和配額	333
服務端點	333
Service Quotas	338
API 限制	339
其他考量	46
發行版本	342
AWS runtime for Apache Spark (emr-spark-8.0.0)	343
AWS runtime for Apache Spark (emr-spark-8.0-預覽)	344
EMR Serverless 7.13.0	345
EMR Serverless 7.12.0	346
EMR Serverless 7.11.0	347
EMR Serverless 7.10.0	347
EMR Serverless 7.9.0	348
EMR Serverless 7.8.0	348
EMR Serverless 7.7.0	348
EMR Serverless 7.6.0	349
EMR Serverless 7.5.0	349
EMR Serverless 7.4.0	349
EMR Serverless 7.3.0	350
EMR Serverless 7.2.0	350
EMR Serverless 7.1.0	351
EMR Serverless 7.0.0	351
EMR Serverless 6.15.0	351
EMR Serverless 6.14.0	352
EMR Serverless 6.13.0	352
EMR Serverless 6.12.0	353
EMR Serverless 6.11.0	353
EMR Serverless 6.10.0	353
EMR Serverless 6.9.0	354

EMR Serverless 6.8.0	355
EMR Serverless 6.7.0	355
引擎特定變更	355
EMR Serverless 6.6.0	356
文件歷史紀錄	358
.....	ccclx

什麼是 Amazon EMR Serverless ？

Amazon EMR Serverless 是 Amazon EMR 的部署選項，可提供無伺服器執行期環境。這可簡化使用最新開放原始碼架構的分析應用程式的操作，例如 Apache Spark 和 Apache Hive。使用 EMR Serverless，您不需要設定、最佳化、保護或操作叢集，即可使用這些架構執行應用程式。

EMR Serverless 可協助您避免資料處理任務過度佈建或佈建不足的資源。EMR Serverless 會自動判斷應用程式所需的資源、取得這些資源來處理任務，並在任務完成時釋出資源。對於應用程式在幾秒鐘內需要回應的使用案例，例如互動式資料分析，您可以在建立應用程式時預先初始化應用程式所需的資源。

透過 EMR Serverless，您可以繼續獲得 Amazon EMR 的優勢，例如開放原始碼相容性、並行，以及適用於熱門架構的最佳化執行期效能。

EMR Serverless 適合希望使用開放原始碼架構輕鬆操作應用程式的客戶。它提供快速的任務啟動、自動容量管理和直接的成本控制。

概念

在本節中，我們會介紹 EMR Serverless 使用者指南中出現的 EMR Serverless 術語和概念。

發行版本

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，您為 EMR Serverless 選擇要部署和設定，以便它們可以執行您的應用程式。當您建立應用程式時，請指定其發行版本。選擇您要在應用程式中使用的 Amazon EMR 發行版本和開放原始碼架構版本。若要進一步了解發行前版本，請參閱 [Amazon EMR Serverless 發行版本](#)。

應用程式

使用 EMR Serverless，您可以建立一或多個使用開放原始碼分析架構的 EMR Serverless 應用程式。若要建立應用程式，請指定下列屬性：

- 您要使用的開放原始碼架構版本的 Amazon EMR 發行版本。若要判斷您的發行版本，請參閱 [Amazon EMR Serverless 發行版本](#)。
- 您希望應用程式使用的特定執行時間，例如 Apache Spark 或 Apache Hive。

建立應用程式後，請將資料處理任務或互動式請求提交至您的應用程式。

每個 EMR Serverless 應用程式在安全 Amazon Virtual Private Cloud (VPC) 上執行，與其他應用程式嚴格不同。此外，使用 AWS Identity and Access Management (IAM) 政策來定義哪些使用者和角色可以存取應用程式。您也可以指定限制來控制和追蹤應用程式產生的用量成本。

當您必須執行下列動作時，請考慮建立多個應用程式：

- 使用不同的開放原始碼架構
- 針對不同的使用案例使用不同版本的開放原始碼架構
- 從一個版本升級到另一個版本時執行 A/B 測試
- 為測試和生產案例維護個別的邏輯環境
- 為不同團隊提供獨立的成本控制和用量追蹤的個別邏輯環境
- 分隔不同的line-of-business應用程式

EMR Serverless 是一項區域服務，可簡化工作負載如何跨區域中的多個可用區域執行。若要進一步了解如何搭配 EMR Serverless 使用應用程式，請參閱 [與 EMR Serverless 應用程式互動並設定](#)。

作業執行

任務執行是提交至 EMR Serverless 應用程式的請求，應用程式會以非同步方式執行並追蹤完成。任務範例包括您提交至 Apache Hive 應用程式的 HiveQL 查詢，或您提交至 Apache Spark 應用程式的 PySpark 資料處理指令碼。提交任務時，您必須指定在 IAM 中撰寫的執行時間角色，該任務會用來存取 AWS 資源，例如 Amazon S3 物件。您可以向應用程式提交多個任務執行請求，而且每個任務執行都可以使用不同的執行時間角色來存取 AWS 資源。EMR Serverless 應用程式在收到任務後立即開始執行任務，並同時執行多個任務請求。若要進一步了解 EMR Serverless 如何執行任務，請參閱 [執行任務](#)。

工作程序

EMR Serverless 應用程式會在內部使用工作者來執行您的工作負載。這些工作者的預設大小取決於您的應用程式類型和 Amazon EMR 發行版本。當您排程任務執行時，請覆寫這些大小。

當您提交任務時，EMR Serverless 會計算應用程式為任務所需的資源，並排程工作者。EMR Serverless 會將工作負載分解為任務、下載映像、佈建和設定工作者，並在任務完成時解除委任。EMR Serverless 會根據任務每個階段所需的工作負載和平行處理，自動擴展或縮減工作者。此自動擴展讓您不需要預估應用程式執行工作負載所需的工作者數量。

預先初始化的容量

EMR Serverless 提供預先初始化的容量功能，可讓工作者在幾秒鐘內初始化並準備好回應。此容量可有效地為應用程式建立工作者的暖集區。若要為每個應用程式設定此功能，請設定應用程式的 `initial-capacity` 參數。當您設定預先初始化的容量時，任務可以立即開始，以便您可以實作迭代應用程式和時間敏感的任務。若要進一步了解預先初始化的工作者，請參閱 [使用 EMR Serverless 時設定應用程式](#)。

EMR Studio

EMR Studio 是用於管理 EMR Serverless 應用程式的使用者主控台。如果在您建立第一個 EMR Serverless 應用程式時，您的帳戶中不存在 EMR Studio，我們會自動為您建立一個。從 Amazon EMR 主控台存取 EMR Studio，或透過 IAM 或 IAM Identity Center 開啟來自身分提供者 (IdP) 的聯合存取。當您這樣做時，使用者可以存取 Studio 和管理 EMR Serverless 應用程式，而無需直接存取 Amazon EMR 主控台。若要進一步了解 EMR Serverless 應用程式如何與 EMR Studio 搭配使用，請參閱 [從 EMR Studio 主控台建立 EMR Serverless 應用程式](#) 和 [從 EMR Studio 主控台執行任務](#)。

EMR Serverless 入門的先決條件

本節說明執行 EMR Serverless 的管理先決條件。這些包括帳戶組態和許可管理。

主題

- [註冊 AWS 帳戶](#)
- [授予許可](#)
- [安裝和設定 AWS CLI](#)
- [開啟 主控台](#)

註冊 AWS 帳戶

若要開始使用 AWS，您需要 AWS 帳戶。如需建立的相關資訊 AWS 帳戶，請參閱《AWS 帳戶管理參考指南》中的 [入門 AWS 帳戶](#)。

授予許可

在生產環境中，我們建議您使用更精細的政策。如需這類政策的範例，請參閱 [EMR Serverless 的使用者存取政策範例](#)。若要進一步了解存取管理，請參閱《IAM 使用者指南》中的 [AWS 資源的存取管理](#)。

對於需要在沙盒環境中開始使用 EMR Serverless 的使用者，請使用類似下列的政策：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EMRStudioCreate",
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:CreateStudioPresignedUrl",
        "elasticmapreduce:DescribeStudio",
        "elasticmapreduce:CreateStudio",
        "elasticmapreduce:ListStudios"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

```

    ]
  },
  {
    "Sid": "EMRServerlessFullAccess",
    "Effect": "Allow",
    "Action": [
      "emr-serverless:*"
    ],
    "Resource": [
      "*"
    ]
  },
  {
    "Sid": "AllowEC2ENICreationWithEMRTags",
    "Effect": "Allow",
    "Action": [
      "ec2:CreateNetworkInterface"
    ],
    "Resource": [
      "arn:aws:ec2:*:*:network-interface/*"
    ],
    "Condition": {
      "StringEquals": {
        "aws:CalledViaLast": "ops.emr-serverless.amazonaws.com"
      }
    }
  },
  {
    "Sid": "AllowEMRServerlessServiceLinkedRoleCreation",
    "Effect": "Allow",
    "Action": [
      "iam:CreateServiceLinkedRole"
    ],
    "Resource": [
      "arn:aws:iam:*:*:role/aws-service-role/*"
    ]
  }
]
}

```

若要提供存取權，請新增權限至您的使用者、群組或角色：

- 中的使用者和群組 AWS IAM Identity Center :

建立權限合集。請按照《AWS IAM Identity Center 使用者指南》中的[建立權限合集](#)說明進行操作。

- 透過身分提供者在 IAM 中管理的使用者：

建立聯合身分的角色。遵循《IAM 使用者指南》的[為第三方身分提供者 \(聯合\) 建立角色](#)中的指示。

- IAM 使用者：
 - 建立您的使用者可擔任的角色。請按照《IAM 使用者指南》的[為 IAM 使用者建立角色](#)中的指示。
 - (不建議) 將政策直接附加至使用者，或將使用者新增至使用者群組。請遵循《IAM 使用者指南》的[新增許可到使用者 \(主控台\)](#)中的指示。

授與程式設計存取權

如果使用者想要與 AWS 外部互動，則需要程式設計存取 AWS 管理主控台。授予程式設計存取權的方式取決於正在存取的使用者類型 AWS。

若要授予使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	到	根據
IAM	(建議) 使用主控台登入資料做為臨時登入資料，以簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	請依照您要使用的介面所提供的指示操作。 <ul style="list-style-type: none"> • 如需 AWS CLI，請參閱 AWS Command Line Interface 《使用者指南》中的登入以進行 AWS 本機開發。 • AWS SDKs 請參閱 AWS SDKs 和工具參考指南中的登入以進行 AWS 本機開發。
人力資源身分 (IAM Identity Center 中管理的使用者)	使用臨時登入資料簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	請依照您要使用的介面所提供的指示操作。 <ul style="list-style-type: none"> • 如需 AWS CLI，請參閱 AWS Command Line Interface 《使用者指南》中的設定

哪個使用者需要程式設計存取權？	到	根據
		<p>AWS CLI 要使用 AWS IAM Identity Center的。</p> <ul style="list-style-type: none"> • AWS SDKs、工具和 AWS APIs，請參閱 AWS SDKs 和工具參考指南中的 IAM Identity Center 身分驗證。
IAM	使用臨時登入資料簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	遵循《IAM 使用者指南》中將 臨時登入資料與 AWS 資源搭配使用 的指示。
IAM	(不建議使用) 使用長期憑證簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	<p>請依照您要使用的介面所提供的指示操作。</p> <ul style="list-style-type: none"> • 如需 AWS CLI，請參閱 AWS Command Line Interface 《使用者指南》中的使用 IAM 使用者憑證進行身分驗證。 • AWS SDKs 和工具，請參閱 AWS SDKs 和工具參考指南中的使用長期憑證進行身分驗證。 • 對於 AWS APIs，請參閱《IAM 使用者指南》中的管理 IAM 使用者的存取金鑰。

安裝和設定 AWS CLI

如果您想要使用 EMR Serverless APIs，請安裝最新版本的 AWS Command Line Interface (AWS CLI)。您不需要從 EMR Studio 主控台 AWS CLI 使用 EMR Serverless，並依照中的步驟在沒有 CLI 的情況下開始使用[從主控台開始使用 EMR Serverless](#)。

若要設定 AWS CLI

1. 若要 AWS CLI 為 macOS、Linux 或 Windows 安裝最新版本的，請參閱[安裝或更新最新版本的 AWS CLI](#)。
2. 若要設定存取的 AWS CLI 和安全設定 AWS 服務，包括 EMR Serverless，請參閱[使用的快速組態aws configure](#)。
3. 若要驗證設定，請在命令提示中輸入下列 DataBrew 命令。

```
aws emr-serverless help
```

AWS CLI 命令會使用 AWS 區域 組態中的預設值，除非您使用參數或設定檔來設定。若要 AWS 區域 使用 參數設定，請將 `--region` 參數新增至每個命令。

若要 AWS 區域 使用設定檔設定您的，請先在 `~/.aws/config` 檔案或 `%UserProfile%/.aws/config` 檔案中新增具名設定檔（適用於 Microsoft Windows）。請遵循 [的具名設定檔 AWS CLI](#) 中的步驟。接著，使用類似下列範例中的命令來設定您的 AWS 區域 和其他設定。

```
[profile emr-serverless]
aws_access_key_id = ACCESS-KEY-ID-OF-IAM-USER
aws_secret_access_key = SECRET-ACCESS-KEY-ID-OF-IAM-USER
region = us-east-1
output = text
```

開啟 主控台

本節中的大多數主控台導向主題都從 [Amazon EMR 主控台](#) 開始。如果您尚未登入您的 AWS 帳戶，請登入，然後開啟 [Amazon EMR 主控台](#) 並繼續前往下一節以繼續開始使用 Amazon EMR。

Amazon EMR Serverless 入門

本教學課程可協助您在部署範例 Spark 或 Hive 工作負載時開始使用 EMR Serverless。您將建立、執行和偵錯自己的應用程式。我們在本教學課程的大部分部分顯示預設選項。

啟動 EMR Serverless 應用程式之前，請先完成下列任務。

主題

- [授予使用 EMR Serverless 的許可](#)
- [準備 EMR Serverless 的儲存體](#)
- [建立 EMR Studio 以執行互動式工作負載](#)
- [建立任務執行期角色](#)
- [從主控台開始使用 EMR Serverless](#)
- [從 入門 AWS CLI](#)

授予使用 EMR Serverless 的許可

若要使用 EMR Serverless，您需要具有連接政策的使用者或 IAM 角色，以授予 EMR Serverless 的許可。若要建立使用者並將適當的政策連接到該使用者，請遵循 [中的指示](#) [授予許可](#)。

準備 EMR Serverless 的儲存體

在本教學課程中，您將使用 S3 儲存貯體來存放使用 EMR Serverless 應用程式執行之範例 Spark 或 Hive 工作負載的輸出檔案和日誌。若要建立儲存貯體，請遵循《Amazon Simple Storage Service 主控台使用者指南》中 [建立儲存貯體](#) 的指示。以新建立的儲存貯 `amzn-s3-demo-bucket` 名稱取代的任何進一步參考。

建立 EMR Studio 以執行互動式工作負載

如果您想要使用 EMR Serverless 透過 EMR Studio 中託管的筆記本執行互動式查詢，您需要指定 S3 儲存貯體和 [EMR Serverless 的最低服務角色](#)，才能建立工作區。如需設定步驟，請參閱《Amazon EMR 管理指南》中的 [設定 EMR Studio](#)。如需互動式工作負載的詳細資訊，請參閱 [透過 EMR Studio 使用 EMR Serverless 執行互動式工作負載](#)。

建立任務執行期角色

EMR Serverless 中的任務執行會使用執行期角色，在執行期提供特定 AWS 服務 和資源的精細許可。在本教學課程中，公有 S3 儲存貯體會託管資料和指令碼。儲存貯體 `amzn-s3-demo-bucket` 存放輸出。

若要設定任務執行期角色，請先使用信任政策建立執行期角色，以便 EMR Serverless 可以使用新角色。接著，將必要的 S3 存取政策連接到該角色。下列步驟會引導您完成此程序。

Console

1. 導覽至位於的 IAM 主控台 <https://console.aws.amazon.com/iam/>。
2. 在左側導覽窗格中選擇 Policies (政策)。
3. 選擇建立政策。
4. 建立新標籤會開啟建立政策頁面。選取政策編輯器做為 Json，並貼上下方的政策 JSON。

Important

將下列政策 `amzn-s3-demo-bucket` 中的 `amzn-s3-demo-bucket` 取代為在中建立的實際儲存貯體名稱 [準備 EMR Serverless 的儲存貯體](#)。這是 S3 存取的基本政策。如需更多任務執行時間角色範例，請參閱 [Amazon EMR Serverless 的任務執行期角色](#)。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadAccessForEMRSamples",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3::*.elasticmapreduce",
        "arn:aws:s3::*.elasticmapreduce/*"
      ]
    }
  ]
}
```

```

    },
    {
      "Sid": "FullAccessToOutputBucket",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ]
    },
    {
      "Sid": "GlueCreateAndReadDataCatalog",
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabase",
        "glue:CreateDatabase",
        "glue:GetDataBases",
        "glue:CreateTable",
        "glue:GetTable",
        "glue:UpdateTable",
        "glue:DeleteTable",
        "glue:GetTables",
        "glue:GetPartition",
        "glue:GetPartitions",
        "glue:CreatePartition",
        "glue:BatchCreatePartition",
        "glue:GetUserDefinedFunctions"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}

```

5. 選擇下一步以輸入政策的名稱，例如 EMRServerlessS3AndGlueAccessPolicy 和 建立政策
6. 在 IAM 主控台 的左側導覽窗格中，選擇角色。

- 選擇建立角色。
- 針對角色類型，選擇自訂信任政策並貼上下列信任政策。這可讓提交至 Amazon EMR Serverless 應用程式的任務 AWS 服務 代表您存取其他。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSTSAssumerole",
      "Effect": "Allow",
      "Principal": {
        "Service": "emr-serverless.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        }
      }
    }
  ]
}
```

- 選擇下一步以導覽至新增許可頁面，然後選擇 EMRServerlessS3AndGlueAccessPolicy。
- 在名稱、檢閱和建立頁面中，針對角色名稱輸入角色的名稱，例如 EMRServerlessS3RuntimeRole。若要建立此 IAM 角色，請選擇建立角色。

CLI

- 建立名為 emr-serverless-trust-policy.json 的檔案，其中包含用於 IAM 角色的信任政策。檔案應包含下列政策。

JSON

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

```

    {
      "Sid": "EMRServerlessTrustPolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": "emr-serverless.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        }
      }
    }
  ]
}

```

2. 建立名為 EMRServerlessS3RuntimeRole 的 IAM 角色。使用您在上一個步驟中建立的信任政策。

```

aws iam create-role \
  --role-name EMRServerlessS3RuntimeRole \
  --assume-role-policy-document file://emr-serverless-trust-policy.json

```

請注意輸出中的 ARN。您可以在任務提交期間使用新角色的 ARN，在此之後稱為 *job-role-arn*。

3. 建立名為的檔案 `emr-sample-access-policy.json`，以定義工作負載的 IAM 政策。這可讓讀取存取存放在公有 S3 儲存貯體中的指令碼和資料，以及讀取寫入存取 *amzn-s3-demo-bucket*。

Important

將以下政策 *amzn-s3-demo-bucket* 中的 取代為在 中建立的實際儲存貯體名稱 [準備 EMR Serverless 的儲存體](#)。這是 Glue AWS 和 S3 存取的基本政策。如需更多任務執行時間角色範例，請參閱 [Amazon EMR Serverless 的任務執行期角色](#)。

JSON

```

{

```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "ReadAccessForEMRSamples",
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:ListBucket"
    ],
    "Resource": [
      "arn:aws:s3::*.elasticmapreduce",
      "arn:aws:s3::*.elasticmapreduce/*"
    ]
  },
  {
    "Sid": "FullAccessToOutputBucket",
    "Effect": "Allow",
    "Action": [
      "s3:PutObject",
      "s3:GetObject",
      "s3:ListBucket",
      "s3:DeleteObject"
    ],
    "Resource": [
      "arn:aws:s3::amzn-s3-demo-bucket",
      "arn:aws:s3:::amzn-s3-demo-bucket/*"
    ]
  },
  {
    "Sid": "GlueCreateAndReadDataCatalog",
    "Effect": "Allow",
    "Action": [
      "glue:GetDatabase",
      "glue:CreateDatabase",
      "glue:GetDataBases",
      "glue:CreateTable",
      "glue:GetTable",
      "glue:UpdateTable",
      "glue>DeleteTable",
      "glue:GetTables",
      "glue:GetPartition",
      "glue:GetPartitions",
      "glue:CreatePartition",
      "glue:BatchCreatePartition",

```

```

        "glue:GetUserDefinedFunctions"
    ],
    "Resource": [
        "*"
    ]
}
]
}

```

4. 使用您在步驟 3 中建立的政策檔案來建立名為 `EMRServerlessS3AndGlueAccessPolicy` 的 IAM 政策。請記下輸出中的 ARN，因為您將在下一個步驟中使用新政策的 ARN。

```

aws iam create-policy \
  --policy-name EMRServerlessS3AndGlueAccessPolicy \
  --policy-document file://emr-sample-access-policy.json

```

在輸出中記下新政策的 ARN。您將在下一個步驟 `policy-arn` 中將其取代為。

5. 將 IAM 政策 `EMRServerlessS3AndGlueAccessPolicy` 連接至任務執行期角色 `EMRServerlessS3RuntimeRole`。

```

aws iam attach-role-policy \
  --role-name EMRServerlessS3RuntimeRole \
  --policy-arn policy-arn

```

從主控台開始使用 EMR Serverless

本節說明使用 EMR Serverless，包括建立 EMR Studio。它還描述了如何提交任務執行和檢視日誌。

要完成的步驟

- [步驟 1：建立 EMR Serverless 應用程式](#)
- [步驟 2：提交任務執行或互動式工作負載](#)
- [步驟 3：檢視應用程式 UI 和日誌](#)
- [步驟 4：清理](#)

步驟 1：建立 EMR Serverless 應用程式

使用 EMR Serverless 建立新的應用程式，如下所示。

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/emr> : // 開啟 Amazon EMR 主控台。
2. 在左側導覽窗格中，選擇 EMR Serverless 以導覽至 EMR Serverless 登陸頁面。
3. 若要建立或管理 EMR Serverless 應用程式，您需要 EMR Studio UI。
 - 如果您在 AWS 區域 要建立應用程式的 中已有 EMR Studio，請選取管理應用程式以導覽至您的 EMR Studio，或選取您要使用的 Studio。
 - 如果您在 AWS 區域 要建立應用程式的 中沒有 EMR Studio，請選擇開始使用，然後選擇建立並啟動 Studio。EMR Serverless 會為您建立 EMR Studio，讓您可以建立和管理應用程式。
4. 在在新標籤中開啟的建立 Studio UI 中，輸入應用程式的名稱、類型和發行版本。如果您只想要執行批次任務，請選取僅對批次任務使用預設設定。針對互動式工作負載，選取使用互動式工作負載的預設設定。您也可以使用此選項在已啟用互動式的應用程式上執行批次任務。如果需要，您可以稍後變更這些設定。

如需詳細資訊，請參閱[建立 Studio](#)。
5. 選取建立應用程式以建立您的第一個應用程式。

繼續下一節[步驟 2：提交任務執行或互動式工作負載](#)以提交任務執行或互動式工作負載。

步驟 2：提交任務執行或互動式工作負載

Spark job run

在本教學課程中，我們使用 PySpark 指令碼來計算多個文字檔案中唯一單字的出現次數。公有的唯讀 S3 儲存貯體會同時存放指令碼和資料集。

執行 Spark 任務

1. 使用下列命令將範例指令碼上傳至您的新儲存貯 `wordcount.py` 體。

```
aws s3 cp s3://us-east-1.elasticmapreduce/emr-containers/samples/wordcount/scripts/wordcount.py s3://amzn-s3-demo-bucket/scripts/
```

2. 完成 [步驟 1：建立 EMR Serverless 應用程式](#) 會帶您前往 EMR Studio 中的應用程式詳細資訊頁面。在那裡，選擇提交任務選項。

3. 在提交任務頁面上，完成以下操作。
 - 在名稱欄位中，輸入您要呼叫任務執行的名稱。
 - 在執行期角色欄位中，輸入您在 [中建立的角色名稱](#) [建立任務執行期角色](#)。
 - 在指令碼位置欄位中，輸入 `s3://amzn-s3-demo-bucket/scripts/wordcount.py` 做為 S3 URI。
 - 在指令碼引數欄位中，輸入 `["s3://amzn-s3-demo-bucket/emr-serverless-spark/output"]`。
 - 在 Spark 屬性區段中，選擇編輯為文字，然後輸入下列組態。

```
--conf spark.executor.cores=1 --conf spark.executor.memory=4g --  
conf spark.driver.cores=1 --conf spark.driver.memory=4g --conf  
spark.executor.instances=1
```

4. 若要開始任務執行，請選擇提交任務。
5. 在任務執行索引標籤中，您應該會看到執行中狀態的新任務執行。

Hive job run

在教學課程的這個部分中，我們會建立資料表、插入一些記錄，以及執行計數彙總查詢。若要執行 Hive 任務，請先建立檔案，其中包含要在單一任務中執行的所有 Hive 查詢、將檔案上傳至 S3，並在啟動 Hive 任務時指定此 S3 路徑。

執行 Hive 任務

1. 建立名為 `hive-query.q1` 的檔案，其中包含您想要在 Hive 任務中執行的所有查詢。

```
create database if not exists emrserverless;  
use emrserverless;  
create table if not exists test_table(id int);  
drop table if exists Values__Tmp__Table__1;  
insert into test_table values (1),(2),(2),(3),(3),(3);  
select id, count(id) from test_table group by id order by id desc;
```

2. 使用下列命令 `hive-query.q1` 上傳至 S3 儲存貯體。

```
aws s3 cp hive-query.q1 s3://amzn-s3-demo-bucket/emr-serverless-hive/query/hive-  
query.q1
```

3. 完成 [步驟 1：建立 EMR Serverless 應用程式](#) 會帶您前往 EMR Studio 中的應用程式詳細資訊頁面。在那裡，選擇提交任務選項。
4. 在提交任務頁面上，完成以下操作。
 - 在名稱欄位中，輸入您要呼叫任務執行的名稱。
 - 在執行期角色欄位中，輸入您在 中建立的角色名稱 [建立任務執行期角色](#)。
 - 在指令碼位置欄位中，輸入 `s3://amzn-s3-demo-bucket/emr-serverless-hive/query/hive-query.q1` 做為 S3 URI。
 - 在 Hive 屬性區段中，選擇編輯為文字，然後輸入下列組態。

```
--hiveconf hive.log.explain.output=false
```

- 在任務組態區段中，選擇編輯為 JSON，然後輸入下列 JSON。

```
{
  "applicationConfiguration":
  [
    {
      "classification": "hive-site",
      "properties": {
        "hive.exec.scratchdir": "s3://amzn-s3-demo-bucket/emr-serverless-hive/hive/scratch",
        "hive.metastore.warehouse.dir": "s3://amzn-s3-demo-bucket/emr-serverless-hive/hive/warehouse",
        "hive.driver.cores": "2",
        "hive.driver.memory": "4g",
        "hive.tez.container.size": "4096",
        "hive.tez.cpu.vcores": "1"
      }
    }
  ]
}
```

5. 若要開始任務執行，請選擇提交任務。
6. 在任務執行索引標籤中，您應該會看到執行中狀態的新任務執行。

Interactive workload

透過 Amazon EMR 6.14.0 及更高版本，您可以使用 EMR Studio 中託管的筆記本來執行 EMR Serverless 中 Spark 的互動式工作負載。如需包含許可和先決條件的詳細資訊，請參閱 [透過 EMR Studio 使用 EMR Serverless 執行互動式工作負載](#)。

建立應用程式並設定必要的許可後，請使用下列步驟透過 EMR Studio 執行互動式筆記本：

1. 導覽至 EMR Studio 中的工作區索引標籤。如果您仍然需要設定 Amazon S3 儲存位置和 [EMR Studio 服務角色](#)，請選取畫面頂端橫幅中的設定 Studio 按鈕。
2. 若要存取筆記本，請選取工作區或建立新的工作區。使用快速啟動在新索引標籤中開啟工作區。
3. 前往新開啟的索引標籤。從左側導覽選取運算圖示。選取 EMR Serverless 做為運算類型。
4. 選取您在上一節中建立的互動式應用程式。
5. 在執行期角色欄位中，輸入 EMR Serverless 應用程式可為任務執行擔任的 IAM 角色名稱。若要進一步了解執行期角色，請參閱《Amazon EMR Serverless 使用者指南》中的[任務執行期角色](#)。
6. 選取連接。這可能需要一分鐘的時間。連接時，頁面會重新整理。
7. 挑選核心並啟動筆記本。您也可以 EMR Serverless 上瀏覽範例筆記本，並將其複製到工作區。若要存取範例筆記本，請導覽至左側導覽中的 {...} 選單，然後瀏覽筆記本檔案名稱 serverless 中具有 的筆記本。
8. 在筆記本中，您可以存取驅動程式日誌連結和 Apache Spark UI 的連結，Apache Spark UI 是一種即時界面，可提供監控任務的指標。如需詳細資訊，請參閱《Amazon [EMR Serverless 使用者指南](#)》中的[監控 EMR Serverless 應用程式和任務](#)。

當您將應用程式連接到 Studio 工作區時，如果應用程式尚未執行，應用程式會自動啟動觸發。您也可以將應用程式連接至工作區之前，預先啟動應用程式並保持就緒狀態。

步驟 3：檢視應用程式 UI 和日誌

若要檢視應用程式 UI，請先識別任務執行。根據任務類型，Spark UI 或 Hive Tez UI 的選項可在該任務執行的第一列選項中使用。選取適當的選項。

如果您選擇 Spark UI，請選擇執行器索引標籤以檢視驅動程式和執行器日誌。如果您選擇 Hive Tez UI，請選擇所有任務索引標籤以檢視日誌。

一旦任務執行狀態顯示為成功，您就可以在 S3 儲存貯體中檢視任務的輸出。

步驟 4：清理

雖然您建立的應用程式應該會在閒置 15 分鐘後自動停止，但仍建議您釋出不打算再次使用的資源。

若要刪除應用程式，請導覽至列出應用程式頁面。選取您建立的應用程式，然後選擇動作 → 停止以停止應用程式。應用程式處於 STOPPED 狀態後，選取相同的應用程式，然後選擇動作 → 刪除。

如需執行 Spark 和 Hive 任務的更多範例，請參閱 [執行 EMR Serverless 任務時使用 Spark 組態](#) 和 [執行 EMR Serverless 任務時使用 Hive 組態](#)。

從入門 AWS CLI

從使用 AWS CLI 命令開始使用 EMR Serverless，以建立應用程式、執行任務、檢查任務執行輸出，以及刪除您的資源。

步驟 1：建立 EMR Serverless 應用程式

使用 [emr-serverless create-application](#) 命令來建立您的第一個 EMR Serverless 應用程式。您需要指定應用程式類型，以及與您要使用的應用程式版本相關聯的 Amazon EMR 發行標籤。應用程式的名稱是選用的。

Spark

若要建立 Spark 應用程式，請執行下列命令。

```
aws emr-serverless create-application \  
  --release-label emr-6.6.0 \  
  --type "SPARK" \  
  --name my-application
```

Hive

若要建立 Hive 應用程式，請執行下列命令。

```
aws emr-serverless create-application \  
  --release-label emr-6.6.0 \  
  --type "HIVE" \  
  --name my-application
```

請注意輸出中傳回的應用程式 ID。您將使用 ID 來啟動應用程式，並在任務提交期間，在此之後稱為 *application-id*。

繼續進行之前 [步驟 2：將任務執行提交到您的 EMR Serverless 應用程式](#)，請確定您的應用程式已使用 [get-application](#) API 達到 CREATED 狀態。

```
aws emr-serverless get-application \  
  --application-id application-id
```

EMR Serverless 會建立工作者來容納您請求的任務。根據預設，這些是隨需建立的，但您也可以在建​​立應用程式時設定 `initialCapacity` 參數來指定預先初始化的容量。您也可以限制應用程式可與 `maximumCapacity` 參數搭配使用的總容量上限。若要進一步了解這些選項，請參閱[使用 EMR Serverless 時設定應用程式](#)。

步驟 2：將任務執行提交到您的 EMR Serverless 應用程式

現在，您的 EMR Serverless 應用程式已準備好執行任務。

Spark

在此步驟中，我們使用 PySpark 指令碼來計算多個文字檔案中唯一單字的出現次數。公有唯讀 S3 儲存貯體會同時存放指令碼和資料集。應用程式會將輸出檔案和日誌資料從 Spark 執行時間傳送至您建立的 S3 儲存貯體中的 `/output` 和 `/logs` 目錄。

執行 Spark 任務

1. 使用下列命令來複製範例指令碼，我們將執行到您的新儲存貯體。

```
aws s3 cp s3://us-east-1.elasticmapreduce/emr-containers/samples/wordcount/  
scripts/wordcount.py s3://amzn-s3-demo-bucket/scripts/
```

2. 在下列命令中，`application-id` 以您的應用程式 ID 取代。`job-role-arn` 使用您在 中建立的執行時間角色 ARN 來取代 [建立任務執行期角色](#)。`job-run-name` 以您想要呼叫任務執行的名稱取代。將所有 `amzn-s3-demo-bucket` 字串取代為您建立的 Amazon S3 儲存貯體，然後 `/output` 新增至路徑。這會在儲存貯體中建立新的資料夾，EMR Serverless 可以在其中複製應用程式的輸出檔案。

```
aws emr-serverless start-job-run \  
  --application-id application-id \  
  --execution-role-arn job-role-arn \  
  --name job-run-name \  
  --job-driver '{  
    "sparkSubmit": {  
      "entryPoint": "s3://amzn-s3-demo-bucket/scripts/wordcount.py",  
      "entryPointArguments": ["s3://amzn-s3-demo-bucket/emr-serverless-  
spark/output"],
```

```

    "sparkSubmitParameters": "--conf spark.executor.cores=1
    --conf spark.executor.memory=4g --conf spark.driver.cores=1 --conf
    spark.driver.memory=4g --conf spark.executor.instances=1"
  }
}'

```

- 請注意輸出中傳回的任務執行 ID。在下列步驟中 *job-run-id* 使用此 ID 取代。

Hive

在本教學課程中，我們會建立資料表、插入一些記錄，以及執行計數彙總查詢。若要執行 Hive 任務，請先建立檔案，其中包含要在單一任務中執行的所有 Hive 查詢、將檔案上傳至 S3，並在您啟動 Hive 任務時指定此 S3 路徑。

執行 Hive 任務

- 建立名為 `hive-query.sql` 的檔案，其中包含您想要在 Hive 任務中執行的所有查詢。

```

create database if not exists emrserverless;
use emrserverless;
create table if not exists test_table(id int);
drop table if exists Values__Tmp__Table__1;
insert into test_table values (1),(2),(2),(3),(3),(3);
select id, count(id) from test_table group by id order by id desc;

```

- 使用下列命令 `hive-query.sql` 上傳至 S3 儲存貯體。

```
aws s3 cp hive-query.sql s3://amzn-s3-demo-bucket/emr-serverless-hive/query/hive-query.sql
```

- 在下列命令中，*application-id* 以您自己的應用程式 ID 取代。*job-role-arn* 使用您在中建立的執行時間角色 ARN 來取代 [建立任務執行期角色](#)。將所有 *amzn-s3-demo-bucket* 字串取代為您建立的 Amazon S3 儲存貯體，並將 `/output` 和 `/logs` 新增至路徑。這會在您的儲存貯體中建立新的資料夾，EMR Serverless 可以在其中複製應用程式的輸出和日誌檔案。

```

aws emr-serverless start-job-run \
  --application-id application-id \
  --execution-role-arn job-role-arn \
  --job-driver '{
    "hive": {
      "query": "s3://amzn-s3-demo-bucket/emr-serverless-hive/query/hive-query.sql",

```

```

        "parameters": "--hiveconf hive.log.explain.output=false"
    }
}' \
--configuration-overrides '{
  "applicationConfiguration": [{
    "classification": "hive-site",
    "properties": {
      "hive.exec.scratchdir": "s3://amzn-s3-demo-bucket/emr-serverless-
hive/hive/scratch",
      "hive.metastore.warehouse.dir": "s3://amzn-s3-demo-bucket/emr-
serverless-hive/hive/warehouse",
      "hive.driver.cores": "2",
      "hive.driver.memory": "4g",
      "hive.tez.container.size": "4096",
      "hive.tez.cpu.vcores": "1"
    }
  ]},
  "monitoringConfiguration": {
    "s3MonitoringConfiguration": {
      "logUri": "s3://amzn-s3-demo-bucket/emr-serverless-hive/logs"
    }
  }
}'

```

4. 請注意輸出中傳回的任務執行 ID。在下列步驟中 *job-run-id* 使用此 ID 取代。

步驟 3：檢閱任務執行的輸出

任務執行通常需要 3-5 分鐘才能完成。

Spark

您可以使用下列命令檢查 Spark 任務的狀態。

```

aws emr-serverless get-job-run \
  --application-id application-id \
  --job-run-id job-run-id

```

將日誌目的地設定為後 `s3://amzn-s3-demo-bucket/emr-serverless-spark/logs`，您可以在下找到此特定任務執行的日誌 `s3://amzn-s3-demo-bucket/emr-serverless-spark/logs/applications/application-id/jobs/job-run-id`。

對於 Spark 應用程式，EMR Serverless 每 30 秒會將事件日誌推送到 S3 日誌目的地中的 `sparklogs` 資料夾。當您的任務完成時，驅動程式和執行器的 Spark 執行期日誌會上傳至工作者類型適當命名的資料夾，例如 `driver` 或 `executor`。PySpark 任務的輸出會上傳至 `s3://amzn-s3-demo-bucket/output/`。

Hive

您可以使用下列命令檢查 Hive 任務的狀態。

```
aws emr-serverless get-job-run \  
  --application-id application-id \  
  --job-run-id job-run-id
```

將日誌目的地設定為 `s3://amzn-s3-demo-bucket/emr-serverless-hive/logs`，您可以在下找到此特定任務執行的日誌 `s3://amzn-s3-demo-bucket/emr-serverless-hive/logs/applications/application-id/jobs/job-run-id`。

對於 Hive 應用程式，EMR Serverless 會持續將 Hive 驅動程式上傳到 S3 日誌目的地的 `HIVE_DRIVER` 資料夾，並將 Tez 任務日誌上傳到 `TEZ_TASK` 資料夾。任務執行達到 `SUCCEEDED` 狀態後，Hive 查詢的輸出會在您在 `monitoringConfiguration` 欄位中指定的 Amazon S3 位置可用 `configurationOverrides`。

步驟 4：清理

完成本教學課程後，請考慮刪除您建立的資源。建議您釋出不打算再次使用的資源。

刪除您的應用程式

若要刪除應用程式，請使用下列命令。

```
aws emr-serverless delete-application \  
  --application-id application-id
```

刪除 S3 日誌儲存貯體

若要刪除 S3 記錄和輸出儲存貯體，請使用下列命令。`amzn-s3-demo-bucket` 將取代為在 中建立之 S3 儲存貯體的實際名稱 [準備 EMR Serverless 的儲存貯體](#)。

```
aws s3 rm s3://amzn-s3-demo-bucket --recursive
```

```
aws s3api delete-bucket --bucket amzn-s3-demo-bucket
```

刪除您的任務執行期角色

若要刪除執行期角色，請從角色分離政策。然後，您可以同時刪除角色和政策。

```
aws iam detach-role-policy \  
  --role-name EMRServerlessS3RuntimeRole \  
  --policy-arn policy-arn
```

若要刪除角色，請使用下列命令。

```
aws iam delete-role \  
  --role-name EMRServerlessS3RuntimeRole
```

若要刪除連接至角色的政策，請使用下列命令。

```
aws iam delete-policy \  
  --policy-arn policy-arn
```

如需執行 Spark 和 Hive 任務的更多範例，請參閱 [執行 EMR Serverless 任務時使用 Spark 組態](#)和 [執行 EMR Serverless 任務時使用 Hive 組態](#)。

與 EMR Serverless 應用程式互動並設定

本節說明如何與 Amazon EMR Serverless 應用程式與 互動 AWS CLI。它還描述應用程式的組態、執行自訂，以及 Spark 和 Hive 引擎的預設值。

主題

- [應用程式狀態](#)
- [從 EMR Studio 主控台建立 EMR Serverless 應用程式](#)
- [在上與您的 EMR Serverless 應用程式互動 AWS CLI](#)
- [使用 EMR Serverless 時設定應用程式](#)
- [自訂 EMR Serverless 映像](#)
- [設定 EMR Serverless 應用程式的 VPC 存取以連線至資料](#)
- [Amazon EMR Serverless 架構選項](#)
- [EMR Serverless 應用程式的任務並行和佇列](#)

應用程式狀態

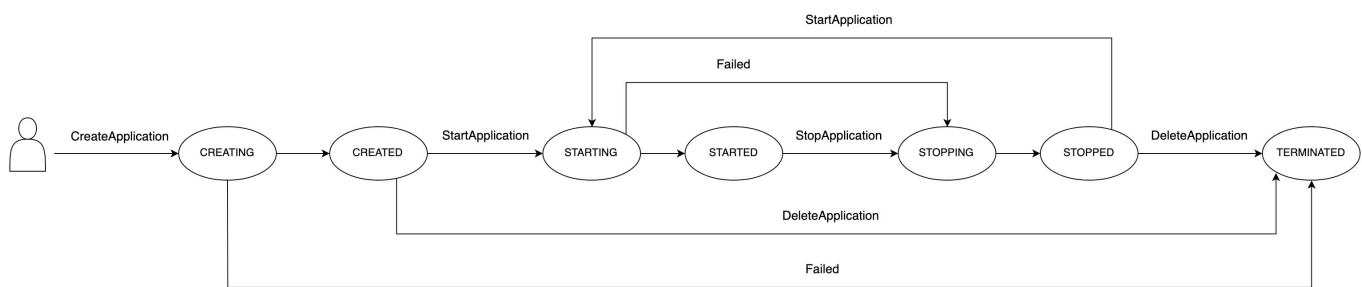
當您使用 EMR Serverless 建立應用程式時，應用程式執行會進入 CREATING 狀態。工作將經過以下狀態，直到其失敗 (以 0 代碼結束) 或失敗 (以與非零代碼結束) 為止。

應用程式可以有下列狀態：

State	Description
正在建立	應用程式正在準備，尚未準備好使用。
已建立	應用程式已建立，但尚未佈建容量。您可以修改應用程式以變更其初始容量組態。
啟動	應用程式正在啟動並佈建容量。
已開始	應用程式已準備好接受新任務。應用程式只有在處於此狀態時，才會接受任務。
Stopping (正在停止)	所有任務都已完成，應用程式正在釋出其容量。

State	Description
已停止	應用程式會停止，而且應用程式上不會執行任何資源。您可以修改應用程式以變更其初始容量組態。
已終止	應用程式已終止，不會顯示在您的應用程式清單中。

下圖說明 EMR Serverless 應用程式狀態的軌跡。



從 EMR Studio 主控台建立 EMR Serverless 應用程式

從 EMR Studio 主控台建立、存取和管理 EMR Serverless 應用程式。若要導覽至 EMR Studio 主控台，請遵循[從主控台入門](#)中的指示。

建立應用程式

使用建立應用程式頁面，依照下列步驟建立 EMR Serverless 應用程式。

1. 在名稱欄位中，輸入您要呼叫應用程式的名稱。
2. 在類型欄位中，選擇 Spark 或 Hive 作為應用程式的類型。
3. 在發行版本欄位中，選擇 EMR 發行編號。
4. 在架構選項中，選擇要使用的指令集架構。如需詳細資訊，請參閱 [Amazon EMR Serverless 架構選項](#)。
 - arm64 — 64 位元 ARM 架構；使用 Graviton 處理器
 - x86_64 — 64 位元 x86 架構；使用以 x86 為基礎的處理器

5. 其餘欄位有兩個應用程式設定選項：預設設定和自訂設定。這些欄位為選用欄位。

預設設定 — 預設設定可讓您使用預先初始化的容量快速建立應用程式。這包括一個驅動程式和一個 Spark 執行器，以及一個驅動程式和一個 Hive Tez 任務。預設設定不會啟用 VPCs 網路連線。應用程式設定為在閒置 15 分鐘時停止，並在任務提交時自動啟動。

自訂設定 — 自訂設定可讓您修改下列屬性。

- **預先初始化容量** — 驅動程式和執行器或 Hive Tez 任務工作者的數量，以及每個工作者的大小。
- **應用程式限制** — 應用程式的最大容量。
- **應用程式行為** — 應用程式的自動啟動和自動停止行為。
- **網路連線** — 網路連線至 VPC 資源。
- **標籤** — 指派給應用程式的自訂標籤。

如需預先初始化容量、應用程式限制和應用程式行為的詳細資訊，請參閱 [使用 EMR Serverless 時設定應用程式](#)。如需網路連線的詳細資訊，請參閱 [設定 EMR Serverless 應用程式的 VPC 存取以連線至資料](#)。

6. 若要建立應用程式，請選擇建立應用程式。

從 EMR Studio 主控台列出應用程式

您可以從列出應用程式頁面存取所有現有的 EMR Serverless 應用程式。您可以選擇應用程式的名稱，以導覽至該應用程式的詳細資訊頁面。

從 EMR Studio 主控台管理應用程式

您可以從列出應用程式頁面或特定應用程式的詳細資訊頁面，對應用程式執行下列動作。

啟動應用程式

選擇此選項以手動啟動應用程式。

停止應用程式

選擇此選項以手動停止應用程式。應用程式必須沒有執行中的任務才能停止。若要進一步了解應用程式狀態轉換，請參閱 [應用程式狀態](#)。

設定應用程式

從設定應用程式頁面編輯應用程式的選用設定。您可以變更大多數應用程式設定。例如，變更應用程式的發行標籤，將其升級至不同版本的 Amazon EMR，或將架構從 x86_64 切換到 arm64。其他選用設定與建立應用程式頁面上自訂設定區段中的設定相同。如需應用程式設定的詳細資訊，請參閱 [建立應用程式](#)。

刪除應用程式

選擇此選項以手動刪除應用程式。您必須停止應用程式才能將其刪除。若要進一步了解應用程式狀態轉換，請參閱 [應用程式狀態](#)。

在上與您的 EMR Serverless 應用程式互動 AWS CLI

從 AWS CLI 建立、描述和刪除個別應用程式。您也可以列出所有應用程式，以便一目了然地存取它們。本節說明如何執行這些動作。如需啟動、停止和應用程式更新等更多應用程式操作，請參閱 [EMR Serverless API 參考](#)。如需如何使用使用 EMR Serverless API 的範例適用於 Java 的 AWS SDK，請參閱 GitHub 儲存庫中的 [Java 範例](#)。如需如何使用使用 EMR Serverless API 的範例適用於 Python (Boto) 的 AWS SDK，請參閱 GitHub 儲存庫中的 [Python 範例](#)。

若要建立應用程式，請使用 `create-application`。您必須指定 SPARK 或 HIVE 做為應用程式 type。此命令會傳回應用程式的 ARN、名稱和 ID。

```
aws emr-serverless create-application \  
--name my-application-name \  
--type 'application-type' \  
--release-label release-version
```

若要描述應用程式，請使用 `get-application` 並提供其 `application-id`。此命令會傳回應用程式的狀態和容量相關組態。

```
aws emr-serverless get-application \  
--application-id application-id
```

若要列出所有應用程式，請呼叫 `list-applications`。此命令會傳回與 `get-application` 相同的屬性，但包含您的所有應用程式。

```
aws emr-serverless list-applications
```

若要刪除您的應用程式，請呼叫 `delete-application` 並提供您的 `application-id`。

```
aws emr-serverless delete-application \  
--application-id application-id
```

使用 EMR Serverless 時設定應用程式

使用 EMR Serverless，設定您使用的應用程式。例如，設定應用程式可擴展的最大容量、設定預先初始化的容量，讓驅動程式和工作者準備好回應，並在應用程式層級指定一組常見的執行時間和監控組態。以下頁面說明如何在使用 EMR Serverless 時設定應用程式。

主題

- [了解 EMR Serverless 中的應用程式行為](#)
- [在 EMR Serverless 中使用應用程式的預先初始化容量](#)
- [EMR Serverless 的預設應用程式組態](#)

了解 EMR Serverless 中的應用程式行為

本節說明任務提交行為、擴展的容量組態，以及 EMR Serverless 的工作者組態設定。

預設應用程式行為

自動啟動 — 根據預設，應用程式設定為在提交任務時自動啟動。您可以關閉此功能。

自動停止 — 根據預設，應用程式會設定為在閒置 15 分鐘時自動停止。當應用程式變更為 STOPPED 狀態時，它會釋出任何已設定的預先初始化容量。您可以修改應用程式自動停止之前的閒置時間，也可以關閉此功能。

容量上限

您可以設定應用程式可擴展的最大容量。您可以指定 CPU、記憶體 (GB) 和磁碟 (GB) 的最大容量。

Note

最佳實務是將您的最大容量設定為與支援的工作者大小成比例，方法是將工作者數量乘以其大小。例如，如果您想要將應用程式限制為 50 個工作者，其中包含 2 個 vCPUs、16 GB 的

記憶體，以及 20 GB 的磁碟，請將最大容量設定為 100 個 vCPUs、800 GB 的記憶體，以及 1000 GB 的磁碟。

支援的工作者組態

下表列出可指定給 EMR Serverless 的支援工作者組態和大小。根據工作負載的需求，為驅動程式和執行器設定不同的大小。

工作者組態和大小

CPU	記憶體	預設暫時性儲存
1 vCPU	最小 2 GB，最大 8 GB，以 1 GB 為增量單位	20 GB - 200 GB
2 vCPU	最小 4 GB，最大 16 GB，以 1 GB 為增量單位	20 GB - 200 GB
4 vCPU	最小 8 GB，最大 30 GB，以 1 GB 為增量單位	20 GB - 200 GB
8 vCPU	最小 16 GB，最大 60 GB，以 4 GB 為單位遞增	20 GB - 200 GB
16 vCPU	最小 32 GB，最大 120 GB，以 8 GB 為增量單位	20 GB - 200 GB

CPU — 每個工作者可以有 1、2、4、8 或 16 vCPUs。

記憶體 — 每個工作者都有記憶體，以 GB 為單位，在先前資料表中列出的限制內。Spark 任務具有記憶體負荷，表示其使用的記憶體超過指定的容器大小。此額外負荷是使用屬性 `spark.driver.memoryOverhead` 和 `spark.executor.memoryOverhead` 來指定。額外負荷的預設值為容器記憶體的 10%，下限為 384 MB。當您選擇工作者大小時，應考慮此額外負荷。

例如，如果您為工作者執行個體選擇 4 vCPUs，且預先初始化的儲存容量為 30 GB，則請為您的 Spark 任務將大約 27 GB 的值設定為執行器記憶體。這可將預先初始化容量的使用率最大化。可用記憶體為 27 GB，加上 27 GB (2.7 GB) 的 10%，總共 29.7 GB。

磁碟：您可以使用大小下限為 20 GB 且上限為 200 GB 的暫存儲存磁碟來設定每個工作者。您只需為超過每個工作者所設定 20 GB 的額外儲存空間付費。

在 EMR Serverless 中使用應用程式的預先初始化容量

EMR Serverless 提供選用功能，可讓驅動程式和工作者在幾秒鐘內預先初始化並準備好回應。這可有效地為應用程式建立工作者的暖集區。此功能稱為預先初始化容量。若要設定此功能，請將應用程式的 `initialCapacity` 參數設定為您要預先初始化的工作者數量。使用預先初始化的工作者容量，任務會立即開始。當您想要實作反覆的應用程式和時間敏感的任务時，這是理想的選擇。

預先初始化容量可讓工作者的暖集區準備好在幾秒鐘內啟動任務和工作階段。即使應用程式閒置，您仍需支付佈建的預先初始化工作者費用，因此我們建議針對受益於快速啟動時間的使用案例啟用它，並調整其大小以獲得最佳資源使用率。EMR Serverless 應用程式會在閒置時自動關閉。建議您在使用預先初始化的工作者時，將此功能維持在開啟狀態，以避免意外費用。

當您提交任務時，如果來自的工作者 `initialCapacity` 可用，任務會使用這些資源來開始執行。如果這些工作者已經由其他任務使用，或者如果任務需要比更多的資源 `initialCapacity`，則應用程式會請求並取得額外的工作者，最高可達應用程式所設定資源的最大限制。當任務完成執行時，它會釋出其使用的工作者，且應用程式可用的資源數目會傳回 `initialCapacity`。即使任務完成執行，應用程式仍會維護資源 `initialCapacity` 的。應用程式會在任務不再需要執行 `initialCapacity` 時，釋出多餘的資源。

應用程式啟動時，預先初始化的容量可供使用。應用程式停止時，預先初始化的容量會變成非作用中。只有當已建立請求的預先初始化容量且可供使用時，應用程式才會移至 `STARTED` 狀態。在應用程式處於 `STARTED` 狀態的整段時間內，EMR Serverless 會保留預先初始化的容量，以供任務或互動式工作負載使用。此功能會還原已發行或失敗容器的容量。這可維持 `InitialCapacity` 參數指定的工作者數量。沒有預先初始化容量的應用程式狀態可以立即從 `CREATED` 變更為 `STARTED`。

如果應用程式在一段時間內未使用，則您可以將應用程式設定為釋放預先初始化的容量，預設值為 15 分鐘。當您提交新任務時，停止的應用程式會自動啟動。您可以在建立應用程式時設定這些自動啟動和停止組態，或在應用程式處於 `CREATED` 或 `STOPPED` 狀態時變更組態。

您可以變更 `InitialCapacity` 計數，並為每個工作者指定運算組態，例如 CPU、記憶體和磁碟。由於您無法進行部分修改，因此請在變更值時指定所有運算組態。您只能在應用程式處於 `CREATED` 或 `STOPPED` 狀態時變更組態。

Note

為了最佳化應用程式對資源的使用，建議您將容器大小與預先初始化的容量工作者大小保持一致。例如，如果您將 Spark 執行器大小設定為 2 CPUs，並將記憶體設定為 8 GB，但您的預先

初始化容量工作者大小為 4 個具有 16 GB 記憶體之 CPU，則 Spark 執行器只會在指派給此任務時使用一半的工作者資源。

自訂 Spark 和 Hive 的預先初始化容量

您可以針對特定大數據架構上執行的工作負載，進一步自訂預先初始化的容量。例如，當工作負載在 Apache Spark 上執行時，請指定有多少工作者開始做為驅動程式，以及有多少工作者開始做為執行器。同樣地，當您使用 Apache Hive 時，請指定有多少工作者開始做為 Hive 驅動程式，以及應執行多少 Tez 任務。

使用預先初始化的容量設定執行 Apache Hive 的應用程式

下列 API 請求會根據 Amazon EMR 發行版本 emr-6.6.0 建立執行 Apache Hive 的應用程式。應用程式從 5 個預先初始化的 Hive 驅動程式開始，每個驅動程式具有 2 個 vCPU 和 4 GB 記憶體，以及 50 個預先初始化的 Tez 任務工作者，每個工作者具有 4 個 vCPU 和 8 GB 記憶體。當 Hive 查詢在此應用程式上執行時，它們會先使用預先初始化的工作者，並立即開始執行。如果所有預先初始化的工作者都忙碌中，並提交了更多 Hive 任務，則應用程式可以擴展到總共 400 個 vCPU 和 1024 GB 的記憶體。您可以選擇性地省略 DRIVER 或 TEZ_TASK 工作者的容量。

```
aws emr-serverless create-application \  
  --type "HIVE" \  
  --name my-application-name \  
  --release-label emr-6.6.0 \  
  --initial-capacity '{  
    "DRIVER": {  
      "workerCount": 5,  
      "workerConfiguration": {  
        "cpu": "2vCPU",  
        "memory": "4GB"  
      }  
    },  
    "TEZ_TASK": {  
      "workerCount": 50,  
      "workerConfiguration": {  
        "cpu": "4vCPU",  
        "memory": "8GB"  
      }  
    }  
  }' \  
  --maximum-capacity '{
```

```
"cpu": "400vCPU",  
"memory": "1024GB"  
}'
```

使用預先初始化的容量設定執行 Apache Spark 的應用程式

下列 API 請求會根據 Amazon EMR 6.6.0 版建立執行 Apache Spark 3.2.0 的應用程式。應用程式從 5 個預先初始化的 Spark 驅動程式開始，每個驅動程式具有 2 個 vCPU 和 4 GB 記憶體，以及 50 個預先初始化的執行器，每個執行器具有 4 個 vCPU 和 8 GB 記憶體。當 Spark 任務在此應用程式上執行時，它們會先使用預先初始化的工作者，並立即開始執行。如果所有預先初始化的工作者都忙碌中，並提交了更多 Spark 任務，則應用程式可以擴展到總共 400 個 vCPU 和 1024 GB 的記憶體。您可以選擇性地省略 DRIVER 或 的容量 EXECUTOR。

Note

Spark 會將具有 10% 預設值的可設定記憶體額外負荷新增至驅動程式和執行器請求的記憶體。若要讓任務使用預先初始化的工作者，初始容量記憶體組態應大於任務和額外負荷請求的記憶體。

```
aws emr-serverless create-application \  
--type "SPARK" \  
--name my-application-name \  
--release-label emr-6.6.0 \  
--initial-capacity '{  
  "DRIVER": {  
    "workerCount": 5,  
    "workerConfiguration": {  
      "cpu": "2vCPU",  
      "memory": "4GB"  
    }  
  },  
  "EXECUTOR": {  
    "workerCount": 50,  
    "workerConfiguration": {  
      "cpu": "4vCPU",  
      "memory": "8GB"  
    }  
  }  
}' \  
--maximum-capacity '{
```

```
"cpu": "400vCPU",  
"memory": "1024GB"  
}'
```

EMR Serverless 的預設應用程式組態

您可以在應用程式層級為您提交的所有任務指定一組通用的執行時間和監控組態。這可減少與需要為每個任務提交相同組態相關聯的額外額外負荷。

您可以在下列時間點修改組態：

- [在提交任務時宣告應用程式層級組態。](#)
- [在任務執行期間覆寫預設組態。](#)

下列各節提供更多詳細資訊和進一步內容的範例。

在應用程式層級宣告組態

您可以為您提交在應用程式下提交的任務指定應用程式層級記錄和執行時間組態屬性。

monitoringConfiguration

若要指定您隨應用程式提交之任務的日誌組態，請使用 [monitoringConfiguration](#) 欄位。如需 EMR Serverless 記錄的詳細資訊，請參閱 [儲存日誌](#)。

runtimeConfiguration

若要指定執行時間組態屬性，例如 spark-defaults，請在欄位中提供組態物件 runtimeConfiguration。這會影響您使用應用程式提交之所有任務的預設組態。如需詳細資訊，請參閱 [Hive 組態覆寫參數](#) 和 [Spark 組態覆寫參數](#)。

可用的組態分類會因特定 EMR Serverless 版本而有所不同。例如，自訂 Log4j spark-driver-log4j2 和 的分類 spark-executor-log4j2 僅適用於 6.8.0 版和更新版本。如需應用程式特定屬性的清單，請參閱 [Spark 任務屬性](#) 和 [Hive 任務屬性](#)。

您也可以 [AWS Secrets Manager 為資料保護](#) 設定 [Apache Log4j2 屬性](#)，並在應用程式層級設定 [Java 17 執行時間](#)。

若要在應用程式層級傳遞 Secrets Manager 秘密，請將下列政策連接至需要使用秘密建立或更新 EMR Serverless 應用程式的使用者和角色。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "SecretsManagerPolicy",
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],
      "Resource": [
        "arn:aws:secretsmanager:us-east-1:123456789012:secret:my-secret-
name-123abc"
      ]
    },
    {
      "Sid": "KMSDecryptPolicy",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012"
      ]
    }
  ]
}
```

如需為秘密建立自訂政策的詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的[適用於的許可政策範例 AWS Secrets Manager](#)。

Note

您在應用程式層級指定的 `runtimeConfiguration` 會映射至 [StartJobRun](#) API `applicationConfiguration` 中的。

範例宣告

下列範例示範如何使用 宣告預設組態create-application。

```
aws emr-serverless create-application \  
  --release-label release-version \  
  --type SPARK \  
  --name my-application-name \  
  --runtime-configuration '[  
    {  
      "classification": "spark-defaults",  
      "properties": {  
        "spark.driver.cores": "4",  
        "spark.executor.cores": "2",  
        "spark.driver.memory": "8G",  
        "spark.executor.memory": "8G",  
        "spark.executor.instances": "2",  
  
        "spark.hadoop.java.jdbc.option.ConnectionDriverName": "org.mariadb.jdbc.Driver",  
        "spark.hadoop.java.jdbc.option.ConnectionURL": "jdbc:mysql://db-host:db-  
port/db-name",  
        "spark.hadoop.java.jdbc.option.ConnectionUserName": "connection-user-  
name",  
        "spark.hadoop.java.jdbc.option.ConnectionPassword":  
        "EMR.secret@SecretID"  
      }  
    },  
    {  
      "classification": "spark-driver-log4j2",  
      "properties": {  
        "rootLogger.level": "error",  
        "logger.IdentifierForClass.name": "classpathForSettingLogger",  
        "logger.IdentifierForClass.level": "info"  
      }  
    }  
  ]' \  
  --monitoring-configuration '{  
    "s3MonitoringConfiguration": {  
      "logUri": "s3://amzn-s3-demo-logging-bucket/logs/app-level"  
    },  
    "managedPersistenceMonitoringConfiguration": {  
      "enabled": false  
    }  
  }
```

```
}'
```

在任務執行期間覆寫組態

您可以使用 [StartJobRun](#) API 指定應用程式組態和監控組態的組態覆寫。然後，EMR Serverless 會合併您在應用程式層級和任務層級指定的組態，以判斷任務執行的組態。

合併發生時的精細程度層級如下：

- [ApplicationConfiguration](#) - 分類類型，例如 spark-defaults。
- [MonitoringConfiguration](#) - 組態類型，例如 s3MonitoringConfiguration。

Note

您在提供的組態優先順序會 [StartJobRun](#) 取代您在應用程式層級提供的組態。

如需詳細資訊優先順序排名，請參閱 [Hive 組態覆寫參數](#) 和 [Spark 組態覆寫參數](#)。

當您啟動任務時，如果您未指定特定組態，則會從應用程式繼承。如果您在任務層級宣告組態，您可以執行下列操作：

- 覆寫現有的組態 - 使用覆寫值在 StartJobRun 請求中提供相同的組態參數。
- 新增其他組態 - 使用您要指定的值，在 StartJobRun 請求中新增新的組態參數。
- 移除現有的組態 - 若要移除應用程式執行期組態，請提供您要移除之組態的金鑰，並傳遞組態 {} 的空白宣告。我們不建議移除包含任務執行所需參數的任何分類。例如，如果您嘗試移除 [Hive 任務所需的屬性](#)，任務將會失敗。

若要移除應用程式監控組態，請使用相關組態類型的適當方法：

- [cloudWatchLoggingConfiguration](#) - 若要移除 cloudWatchLogging，請將啟用的旗標傳遞為 false。
- [managedPersistenceMonitoringConfiguration](#) - 若要移除受管持久性設定並回到預設啟用狀態，請傳遞組態 {} 的空白宣告。
- [s3MonitoringConfiguration](#) - 若要移除 s3MonitoringConfiguration，請傳遞組態 {} 的空白宣告。

覆寫範例

下列範例顯示您可以在 提交任務期間執行的不同操作start-job-run。

```
aws emr-serverless start-job-run \
  --application-id your-application-id \
  --execution-role-arn your-job-role-arn \
  --job-driver '{
    "sparkSubmit": {
      "entryPoint": "s3://us-east-1.elasticmapreduce/emr-containers/samples/
wordcount/scripts/wordcount.py",
      "entryPointArguments": ["s3://amzn-s3-demo-destination-bucket1/
wordcount_output"]
    }
  }' \
  --configuration-overrides '{
    "applicationConfiguration": [
      {
        // Override existing configuration for spark-defaults in the
application
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.cores": "2",
          "spark.executor.cores": "1",
          "spark.driver.memory": "4G",
          "spark.executor.memory": "4G"
        }
      },
      {
        // Add configuration for spark-executor-log4j2
        "classification": "spark-executor-log4j2",
        "properties": {
          "rootLogger.level": "error",
          "logger.IdentifierForClass.name": "classpathForSettingLogger",
          "logger.IdentifierForClass.level": "info"
        }
      },
      {
        // Remove existing configuration for spark-driver-log4j2 from the
application
        "classification": "spark-driver-log4j2",
        "properties": {}
      }
    ],
  }
```

```
"monitoringConfiguration": {
  "managedPersistenceMonitoringConfiguration": {
    // Override existing configuration for managed persistence
    "enabled": true
  },
  "s3MonitoringConfiguration": {
    // Remove configuration of S3 monitoring
  },
  "cloudWatchLoggingConfiguration": {
    // Add configuration for CloudWatch logging
    "enabled": true
  }
}
```

在任務執行時，將根據 [和](#) 中所述的優先順序覆寫排名套用下列分類 [Hive 組態覆寫參數](#) 和組態 [Spark 組態覆寫參數](#)。

- 分類 `spark-defaults` 將使用任務層級指定的屬性進行更新。此分類 `StartJobRun` 只會考慮 `中` 包含的屬性。
- 分類 `spark-executor-log4j2` 將新增至現有的分類清單中。
- `spark-driver-log4j2` 將會移除分類。
- 的組態 `managedPersistenceMonitoringConfiguration` 將以任務層級的組態更新。
- 的組態 `s3MonitoringConfiguration` 將被移除。
- 的組態 `cloudWatchLoggingConfiguration` 會新增至現有的監控組態。

自訂 EMR Serverless 映像

從 Amazon EMR 6.9.0 開始，使用自訂映像將應用程式相依性和執行期環境封裝到具有 Amazon EMR Serverless 的單一容器中。這可簡化您管理工作負載相依性的方式，並讓套件更具可攜性。當您自訂 EMR Serverless 映像時，可提供下列優點：

- 安裝和設定針對工作負載最佳化的套件。這些套件並未廣泛用於 Amazon EMR 執行期環境的公有分佈。
- 將 EMR Serverless 與組織中目前建立的建置、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，例如映像掃描，以符合組織內的合規和管理要求。
- 可讓您為應用程式使用自己的 JDK 和 Python 版本。

EMR Serverless 會在您建立自己的映像時，提供使用做為基礎的映像。基礎映像為映像提供與 EMR Serverless 互動的基本 jar、組態和程式庫。您可以在 [Amazon ECR Public Gallery](#) 中找到基礎映像。使用符合您應用程式類型 (Spark 或 Hive) 和發行版本的映像。例如，如果您在 Amazon EMR 6.9.0 版上建立應用程式，請使用下列映像。

Type	影像
Spark	public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest
Hive	public.ecr.aws/emr-serverless/hive/emr-6.9.0:latest

先決條件

建立 EMR Serverless 自訂映像之前，請先完成這些先決條件。

1. 在用於啟動 EMR Serverless 應用程式的相同 中建立 AWS 區域 Amazon ECR 儲存庫。若要建立 Amazon ECR 私有儲存庫，請參閱[建立私有儲存庫](#)。
2. 若要授予使用者存取 Amazon ECR 儲存庫的權限，請將下列政策新增至使用此儲存庫中的映像建立或更新 EMR Serverless 應用程式的使用者和角色。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ECRRepositoryListGetPolicy",
      "Effect": "Allow",
      "Action": [
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "ecr:DescribeImages"
      ],
      "Resource": [
        "arn:aws:ecr:*:123456789012:repository/my-repo"
      ]
    }
  ]
}
```

```
}
```

如需 Amazon ECR 身分型政策的更多範例，請參閱 [Amazon Elastic Container Registry 身分型政策範例](#)。

步驟 1：從 EMR Serverless 基礎映像建立自訂映像

首先，建立以使用您偏好基礎映像的 FROM 指令開頭的 [Dockerfile](#)。在 FROM 指示之後，包含您要對影像進行的任何修改。基礎映像會自動將 USER 設定為 `hadoop`。此設定沒有您包含的所有修改的許可。作為解決方法，請將 USER 設定為 `root`，修改您的映像，然後將 USER 回 `hadoop:hadoop`。若要參考常見使用案例的範例，請參閱 [搭配 EMR Serverless 使用自訂映像](#)。

```
# Dockerfile
FROM public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest

USER root
# MODIFICATIONS GO HERE

# EMRS runs the image as hadoop
USER hadoop:hadoop
```

在您擁有 Dockerfile 之後，請使用下列命令建置映像。

```
# build the docker image
docker build . -t aws-account-id.dkr.ecr.region.amazonaws.com/my-repository[:tag]or[@digest]
```

步驟 2：在本機驗證映像

EMR Serverless 提供離線工具，可靜態檢查自訂映像，以驗證基本檔案、環境變數和正確的映像組態。如需如何安裝和執行工具的資訊，請參閱 [Amazon EMR Serverless Image CLI GitHub](#)。

安裝工具後，請執行下列命令來驗證映像：

```
amazon-emr-serverless-image \
validate-image -r emr-6.9.0 -t spark \
-i aws-account-id.dkr.ecr.region.amazonaws.com/my-repository:tag/@digest
```

輸出如下所示。

```
Amazon EMR Serverless - Image CLI
Version: 0.0.1
... Checking if docker cli is installed
... Checking Image Manifest
[INFO] Image ID: 9e2f4359cf5beb466a8a2ed047ab61c9d37786c555655fc122272758f761b41a
[INFO] Created On: 2022-12-02T07:46:42.586249984Z
[INFO] Default User Set to hadoop:hadoop : PASS
[INFO] Working Directory Set to : PASS
[INFO] Entrypoint Set to /usr/bin/entrypoint.sh : PASS
[INFO] HADOOP_HOME is set with value: /usr/lib/hadoop : PASS
[INFO] HADOOP_LIBEXEC_DIR is set with value: /usr/lib/hadoop/libexec : PASS
[INFO] HADOOP_USER_HOME is set with value: /home/hadoop : PASS
[INFO] HADOOP_YARN_HOME is set with value: /usr/lib/hadoop-yarn : PASS
[INFO] HIVE_HOME is set with value: /usr/lib/hive : PASS
[INFO] JAVA_HOME is set with value: /etc/alternatives/jre : PASS
[INFO] TEZ_HOME is set with value: /usr/lib/tez : PASS
[INFO] YARN_HOME is set with value: /usr/lib/hadoop-yarn : 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 hadoop-yarn-jars in /usr/lib/hadoop-yarn: PASS
[INFO] File Structure Test for hive-bin-files in /usr/bin: PASS
[INFO] File Structure Test for hive-jars in /usr/lib/hive/lib: PASS
[INFO] File Structure Test for java-bin in /etc/alternatives/jre/bin: PASS
[INFO] File Structure Test for tez-jars in /usr/lib/tez: PASS
-----
Overall Custom Image Validation Succeeded.
-----
```

步驟 3：將映像上傳至您的 Amazon ECR 儲存庫

使用下列命令將您的 Amazon ECR 映像推送到您的 Amazon ECR 儲存庫。請確定您擁有將映像推送至儲存庫的正確 IAM 許可。如需詳細資訊，請參閱《Amazon ECR 使用者指南》中的[推送映像](#)。

```
# login to ECR repo
aws ecr get-login-password --region region | docker login --username AWS --password-
stdin aws-account-id.dkr.ecr.region.amazonaws.com

# push the docker image
docker push aws-account-id.dkr.ecr.region.amazonaws.com/my-repository:tag/@digest
```

步驟 4：使用自訂映像建立或更新應用程式

根據您要啟動應用程式的方式選擇 AWS 管理主控台 索引標籤或 AWS CLI 索引標籤，然後完成下列步驟。

Console

1. 登入 EMR Studio 主控台，網址為 <https://console.aws.amazon.com/emr>。導覽至您的應用程式，或使用建立應用程式中的指示 [建立新的應用程式](#)。
2. 若要在建立或更新 EMR Serverless 應用程式時指定自訂映像，請在應用程式設定選項中選取自訂設定。
3. 在自訂映像設定區段中，選取搭配此應用程式使用自訂映像核取方塊。
4. 將 Amazon ECR 映像 URI 貼入映像 URI 欄位。EMR Serverless 會將此映像用於應用程式的所有工作者類型。或者，您可以選擇不同的自訂映像，並為每個工作者類型貼上不同的 Amazon ECR URIs。

CLI

- 使用 `image-configuration` 參數建立應用程式。EMR Serverless 會將此設定套用至所有工作者類型。

```
aws emr-serverless create-application \  
--release-label emr-6.9.0 \  
--type SPARK \  
--image-configuration '{  
    "imageUri": "aws-account-id.dkr.ecr.region.amazonaws.com/my-repository:tag/  
@digest"  
}'
```

若要為每個工作者類型建立具有不同映像設定的應用程式，請使用 `worker-type-specifications` 參數。

```
aws emr-serverless create-application \  
--release-label emr-6.9.0 \  
--type SPARK \  
--worker-type-specifications '{  
    "Driver": {  
        "imageConfiguration": {
```

```

        "imageUri": "aws-account-id.dkr.ecr.region.amazonaws.com/my-
repository:tag/@digest"
    }
  },
  "Executor" : {
    "imageConfiguration": {
      "imageUri": "aws-account-id.dkr.ecr.region.amazonaws.com/my-
repository:tag/@digest"
    }
  }
}'

```

若要更新應用程式，請使用 `image-configuration` 參數。EMR Serverless 會將此設定套用至所有工作者類型。

```

aws emr-serverless update-application \
--application-id application-id \
--image-configuration '{
  "imageUri": "aws-account-id.dkr.ecr.region.amazonaws.com/my-repository:tag/
@digest"
}'

```

步驟 5：允許 EMR Serverless 存取自訂映像儲存庫

將下列資源政策新增至 Amazon ECR 儲存庫，以允許 EMR Serverless 服務主體使用此儲存庫的 `get`、`describe` 和 `download` 請求。

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EmrServerlessCustomImageSupport",
      "Effect": "Allow",
      "Principal": {
        "Service": "emr-serverless.amazonaws.com"
      },
      "Action": [
        "ecr:BatchGetImage",

```

```

    "ecr:DescribeImages",
    "ecr:GetDownloadUrlForLayer"
  ],
  "Resource": "arn:aws:ecr:*:123456789012:repository/my-repo",
  "Condition": {
    "ArnLike": {
      "aws:SourceArn": "arn:aws:emr-serverless:*:123456789012:/applications/
*"
    }
  }
}
]
}

```

做為安全最佳實務，請將 `aws:SourceArn` 條件金鑰新增至儲存庫政策。IAM 全域條件金鑰 `aws:SourceArn` 可確保 EMR Serverless 僅針對應用程式 ARN 使用儲存庫。如需 Amazon ECR 儲存庫政策的詳細資訊，請參閱 [建立私有儲存庫](#)。

考量和限制

當您使用自訂映像時，請考慮下列事項：

- 使用符合應用程式類型 (Spark 或 Hive) 和發行標籤 (例如 `emr-6.9.0`) 的正確基礎映像。
- EMR Serverless 會忽略 Docker 檔案中的 `[CMD]` 或 `[ENTRYPOINT]` 指示。使用 Docker 檔案中的常見指示，例如 `[RUN]`、`[COPY]` 和 `[WORKDIR]`。
- 建立自訂映像 `TEZ_HOME` 時 `JAVA_HOME`，請勿修改環境變數 `SPARK_HOME`、`HIVE_HOME`、。
- 自訂映像的大小不得超過 10 GB。
- 如果您在 Amazon EMR 基礎映像中修改二進位檔或 jar，這可能會導致應用程式或任務啟動失敗。
- Amazon ECR 儲存庫必須與您用來啟動 EMR Serverless 應用程式 AWS 區域 的位置相同。

設定 EMR Serverless 應用程式的 VPC 存取以連線至資料

您可以設定 EMR Serverless 應用程式連線到 VPC 內的資料存放區，例如 Amazon Redshift 叢集、Amazon RDS 資料庫或具有 VPC 端點的 Amazon S3 儲存貯體。您的 EMR Serverless 應用程式對 VPC 內的資料存放區具有傳出連線。根據預設，EMR Serverless 會封鎖對應用程式的傳入存取和傳出網際網路存取，以增強安全性。

Note

如果您想要為應用程式使用外部 Hive 中繼存放區資料庫，則必須設定 VPC 存取。如需如何設定外部 Hive 中繼存放區的資訊，請參閱[中繼存放區組態](#)。

建立應用程式

在建立應用程式頁面上，選擇自訂設定，並指定 EMR Serverless 應用程式可以使用的 VPC、子網路和安全群組。

VPC

選擇包含您的資料存放區的虛擬私有雲端 (VPC) 名稱。建立應用程式頁面會列出所選的所有 VPCs AWS 區域。

子網路

選擇 VPC 內包含資料存放區子網路。建立應用程式頁面會列出 VPC 中資料存放區的所有子網路。同時支援公有和私有子網路。您可以將私有或公有子網路傳遞給應用程式。選擇是否擁有公有或私有子網路有幾個需要注意的相關考量。

針對私有子網路：

- 相關聯的路由表不得有網際網路閘道。
- 對於網際網路的傳出連線，如有需要，請使用 NAT Gateway 設定傳出路由。若要設定 NAT 閘道，請參閱[NAT 閘道](#)。
- 對於 Amazon S3 連線，請設定 NAT Gateway 或 VPC 端點。若要設定 S3 VPC 端點，請參閱[建立閘道端點](#)。
- 如果您設定 S3 VPC 端點，並連接端點政策來控制存取，請遵循[使用受管儲存體記錄 EMR Serverless](#) 中的指示，為 EMR Serverless 提供儲存和提供應用程式日誌的許可。
- 若要連線至 VPC AWS 服務以外的其他，例如 Amazon DynamoDB，請設定 VPC 端點或 NAT 閘道。若要設定的 VPC 端點 AWS 服務，請參閱[使用 VPC 端點](#)。

Note

當您在私有子網路中設定 Amazon EMR Serverless 應用程式時，我們建議您也為 Amazon S3 設定 VPC 端點。如果您的 EMR Serverless 應用程式位於沒有 Amazon S3 VPC 端點的私

有子網路中，則會產生與 S3 流量相關聯的額外 NAT 閘道費用。這是因為未設定 VPC 端點時，EMR 應用程式和 Amazon S3 之間的流量不會保留在您的 VPC 內。

針對公有子網路：

- 這些有網際網路閘道的路由。
- 您必須確保適當的安全群組組態，以控制傳出流量。

工作者可以透過傳出流量連接到 VPC 內的資料存放區。根據預設，EMR Serverless 會封鎖對工作者的傳入存取。這是為了提升安全性。

當您使用時 AWS Config，EMR Serverless 會為每個工作者建立彈性網路界面項目記錄。若要避免與此資源相關的成本，請考慮關閉 `AWS::EC2::NetworkInterface` AWS Config。

Note

我們建議您跨多個可用區域選取多個子網路。這是因為您選擇的子網路會決定可供 EMR Serverless 應用程式啟動的可用區域。每個工作者都會在啟動的子網路上使用 IP 地址。請確定指定的子網路有足夠的 IP 地址，可供您計劃啟動的工作者數量使用。如需子網路規劃的詳細資訊，請參閱 [the section called “子網路規劃的最佳實務”](#)。

子網路的考量和限制

- 具有公有子網路的 EMR Serverless 不支援 AWS Lake Formation。
- 公有子網路不支援傳入流量。

Security groups (安全群組)

選擇一或多個可與資料存放區通訊的安全群組。建立應用程式頁面會列出 VPC 中的所有安全群組。EMR Serverless 會將這些安全群組與連接到 VPC 子網路的彈性網路介面建立關聯。

Note

我們建議您為 EMR Serverless 應用程式建立個別的安全群組。如果安全群組的連接埠在 `0.0.0.0/0` 或 `::/0` 範圍開放給公有網際網路，EMR Serverless 不允許您建立/更新/啟動應用

程式。這可提供增強的安全性、隔離功能，並讓管理網路規則更有效率。例如，這會封鎖對具有公有 IP 地址的工作者的意外流量。例如，若要與 Amazon Redshift 叢集通訊，請定義 Redshift 和 EMR Serverless 安全群組之間的流量規則，如下節範例所示。

Example範例 — 與 Amazon Redshift 叢集通訊

1. 從其中一個 EMR Serverless 安全群組將傳入流量規則新增至 Amazon Redshift 安全群組。

Type	通訊協定	連接埠範圍	來源
所有 TCP	TCP	5439	emr-serverless-security-group

2. 新增來自其中一個 EMR Serverless 安全群組的傳出流量規則。以兩種方式之一執行此操作。首先，開啟所有連接埠的傳出流量。

Type	通訊協定	連接埠範圍	目標
所有流量	TCP	ALL	0.0.0.0/0

或者，您可以將傳出流量限制為 Amazon Redshift 叢集。只有在應用程式必須與 Amazon Redshift 叢集通訊時，這才很有用。

Type	通訊協定	連接埠範圍	來源
所有 TCP	TCP	5439	redshift-security-group

設定應用程式

您可以從設定應用程式頁面變更現有 EMR Serverless 應用程式的網路組態。

存取任務執行詳細資訊

在任務執行詳細資訊頁面上，存取任務用於特定執行的子網路。請注意，任務只會在從指定子網路中選取的一個子網路中執行。

子網路規劃的最佳實務

AWS 資源是在子網路中建立，子網路是 Amazon VPC 中可用 IP 地址的子集。例如，具有 /16 網路遮罩的 VPC 最多有 65,536 個可用的 IP 地址，可以使用子網路遮罩分成多個較小的網路。例如，您可以將此範圍分割為兩個子網路，每個子網路使用 /17 遮罩和 32,768 個可用的 IP 地址。子網路位於可用區域內，無法跨區域。

子網路的設計應謹記您的 EMR Serverless 應用程式擴展限制。例如，如果您的應用程式請求 4 個 vCpu 工作者，並且可擴展到 4,000 個 vCpu，則您的應用程式最多需要 1,000 個工作者，總共需要 1,000 個網路介面。我們建議您跨多個可用區域建立子網路。這可讓 EMR Serverless 在可用區域故障時，在極少數情況下重試您的任務，或在不同的可用區域中佈建預先初始化的容量。因此，至少兩個可用區域中的每個子網路應具有超過 1,000 個可用的 IP 地址。

您需要遮罩大小小於或等於 22 的子網路，才能佈建 1,000 個網路介面。任何大於 22 的遮罩都不符合要求。例如，/23 的子網路遮罩提供 512 個 IP 地址，而 /22 的遮罩提供 1024，而 /21 的遮罩提供 2048 個 IP 地址。以下是 /16 網路遮罩 VPC 中 /22 遮罩可配置給不同可用區域的 4 個子網路範例。可用和可用的 IP 地址之間有五個差異，因為每個子網路中的前四個 IP 地址和最後一個 IP 地址是由保留 AWS。

子網路 ID	子網路地址	子網路遮罩	IP 地址範圍	可用的 IP 地址	可用的 IP 地址
1	10.0.0.0	255.255.252.0/22	10.0.0.0 - 10.0.3.255	1,024	1,019
2	10.0.4.0	255.255.252.0/22	10.0.4.0 - 10.0.7.255	1,024	1,019
3	10.0.8.0	255.255.252.0/22	10.0.8.0 - 10.0.11.255	1,024	1,019
4	10.0.12.0	255.255.252.0/22	10.0.12.0 - 10.0.15.255	1,024	1,019

您應該評估工作負載是否適合較大的工作者大小。使用較大的工作者大小需要的網路界面較少。例如，使用應用程式擴展限制為 4,000 vCpu 的 16vCpu 工作者，最多需要 250 個工作者，總共需要 250 個可用的 IP 地址來佈建網路介面。vCpu 您需要遮罩大小小於或等於 24 的多個可用區域中的子網路，才能佈建 250 個網路介面。任何大於 24 的遮罩大小都會提供少於 250 個 IP 地址。

如果您跨多個應用程式共用子網路，則每個子網路的設計應謹記所有應用程式的集體擴展限制。例如，如果您有 3 個應用程式請求 4 個 vCpu 工作者，且每個應用程式可擴展至 4000 個 vCpu，並具有 12,000 個 vCpu 帳戶層級服務配額，則每個子網路都需要 3000 個可用的 IP 地址。如果要使用的 VPC 沒有足夠數目的 IP 地址，請嘗試增加可用 IP 地址的數目。您可以透過將其他無類別域間路由 (CIDR) 區塊與 VPC 關聯來完成此操作。如需詳細資訊，請參閱《Amazon VPC [使用者指南](#)》中的[將其他 IPv4 CIDR 區塊與 VPC 建立關聯](#)。

您可以使用線上提供的許多工具之一，快速產生子網路定義並檢閱其可用的 IP 地址範圍。

Amazon EMR Serverless 架構選項

Amazon EMR Serverless 應用程式的指示集架構會決定應用程式用來執行任務的處理器類型。Amazon EMR 為您的應用程式提供兩種架構選項：x86_64 和 arm64。EMR Serverless 會在最新一代的執行個體可用時自動更新，因此您的應用程式可以使用較新的執行個體，而無需您付出額外的努力。

主題

- [使用 x86_64 架構](#)
- [使用 arm64 架構 \(Graviton\)](#)
- [使用 Graviton 支援啟動新應用程式](#)
- [設定現有應用程式以使用 Graviton](#)
- [使用 Graviton 時的考量事項](#)

使用 x86_64 架構

x86_64 架構也稱為 x86 64 位元或 x64。x86_64 是 EMR Serverless 應用程式的預設選項。此架構使用 x86 型處理器，並與大多數第三方工具和程式庫相容。

大多數應用程式都與 x86 硬體平台相容，並且可以在預設 x86_64 架構上成功執行。不過，如果您的應用程式與 64 位元 ARM 相容，請切換到 arm64 以使用 Graviton 處理器來改善效能、運算能力和記憶體。相較於在 x86 架構上執行大小相等的執行個體，在 arm64 架構上執行執行個體的成本較低。

使用 arm64 架構 (Graviton)

AWS Graviton 處理器是由 AWS 使用 64 位元 ARM Neoverse 核心自訂設計，並利用 arm64 架構（也稱為 Arch64 或 64 位元 ARM）。EMR Serverless AWS 上可用的 Graviton 處理器系列包括 Graviton3 和 Graviton2 處理器。相較於在 x86_64 架構上執行的同等工作負載，這些處理器可為 Spark 和 Hive 工作負載提供卓越的價格效能。當可用時，EMR Serverless 會自動使用最新一代的處理器，而無須費力升級到最新一代的處理器。

使用 Graviton 支援啟動新應用程式

使用下列其中一種方法來啟動使用 arm64 架構的應用程式。

AWS CLI

若要從使用 Graviton 處理器啟動應用程式 AWS CLI，請在 create-application API 中指定 ARM64 作為 architecture 參數。在其他參數中為您的應用程式提供適當的值。

```
aws emr-serverless create-application \  
  --name my-graviton-app \  
  --release-label emr-6.8.0 \  
  --type "SPARK" \  
  --architecture "ARM64" \  
  --region us-west-2
```

EMR Studio

若要從 EMR Studio 使用 Graviton 處理器啟動應用程式，請在建立或更新應用程式時選擇 arm64 作為架構選項。

設定現有應用程式以使用 Graviton

您可以將現有的 Amazon EMR Serverless 應用程式設定為搭配 SDK AWS CLI 或 EMR Studio 使用 Graviton (arm64) 架構。

將現有應用程式從 x86 轉換為 arm64

1. 確認您使用的是支援 architecture 參數的 [AWS CLI/SDK](#) 最新主要版本。
2. 確認沒有任務正在執行，然後停止應用程式。

```
aws emr-serverless stop-application \  
  --name my-graviton-app
```

```
--application-id application-id \  
--region us-west-2
```

- 若要更新應用程式以使用 Graviton，請在 update-application API 中 ARM64 為 architecture 參數指定。

```
aws emr-serverless update-application \  
--application-id application-id \  
--architecture 'ARM64' \  
--region us-west-2
```

- 若要驗證應用程式的 CPU 架構現在是 ARM64，請使用 get-application API。

```
aws emr-serverless get-application \  
--application-id application-id \  
--region us-west-2
```

- 當您準備好時，請重新啟動應用程式。

```
aws emr-serverless start-application \  
--application-id application-id \  
--region us-west-2
```

使用 Graviton 時的考量事項

在使用 arm64 for Graviton 支援啟動 EMR Serverless 應用程式之前，請確認下列事項。

程式庫相容性

當您選取 Graviton (arm64) 做為架構選項時，請確定第三方套件和程式庫與 64 位元 ARM 架構相容。如需有關如何將 Python 程式庫封裝到與您所選架構相容的 Python 虛擬環境中的資訊，請參閱 [搭配 EMR Serverless 使用 Python 程式庫](#)。

若要進一步了解，請參閱 GitHub 上的 [AWS Graviton 入門](#) 儲存庫。此儲存庫包含可協助您開始使用 ARM 型 Graviton 的基本資源。

EMR Serverless 應用程式的任務並行和佇列

從 Amazon EMR 7.0.0 版及更新版本開始，為您的應用程式指定任務執行佇列逾時和並行組態。當您指定此組態時，Amazon EMR Serverless 會從佇列任務開始，並根據應用程式的並行使用率開始執

行。例如，如果您的任務執行並行為 10，您的應用程式一次只會執行十個任務。剩餘的任務會排入佇列，直到其中一個執行中的任務終止為止。如果提早達到佇列逾時，您的任務會逾時。如需詳細資訊，請參閱 [任務執行狀態](#)。

並行和佇列的主要優點

需要提交許多任務時，任務並行和佇列可提供下列優點：

- 它有助於控制並行執行任務，以有效率地使用您的應用程式層級容量限制。
- 佇列可以包含突增的任務提交，並具有可設定的逾時設定。

並行和佇列入門

下列程序示範實作並行和佇列的幾種不同方式。

使用 AWS CLI

1. 建立具有佇列逾時和並行任務執行的 Amazon EMR Serverless 應用程式：

```
aws emr-serverless create-application \  
--release-label emr-7.0.0 \  
--type SPARK \  
--scheduler-configuration '{"maxConcurrentRuns": 1, "queueTimeoutMinutes": 30}'
```

2. 更新應用程式以變更任務佇列逾時和並行：

```
aws emr-serverless update-application \  
--application-id application-id \  
--scheduler-configuration '{"maxConcurrentRuns": 5, "queueTimeoutMinutes": 30}'
```

Note

您可以更新現有的應用程式，以啟用任務並行和佇列。若要這樣做，應用程式必須具有發行標籤 `emr-7.0.0` 或更新版本。

使用 AWS 管理主控台

下列步驟示範如何使用 開始使用任務並行和佇列 AWS 管理主控台：

1. 前往 EMR Studio 並選擇建立具有發行標籤 EMR-7.0.0 或更新版本的應用程式。
2. 在應用程式設定選項下，選取使用自訂設定選項。
3. 在其他組態下，有任務執行設定的區段。選取啟用任務並行選項以啟用此功能。
4. 選擇後，選取並行任務執行和佇列逾時，分別設定並行任務執行和佇列逾時的數量。如果您未輸入這些設定的值，則會使用預設值。
5. 選擇建立應用程式，則會在啟用此功能的情況下建立應用程式。若要驗證，請前往儀表板，選取您的應用程式，然後在屬性索引標籤下檢查，以判斷功能是否已啟用。

在組態之後，提交啟用此功能的任務。

並行和佇列的考量事項

當您實作並行和佇列時，請考慮下列事項：

- Amazon EMR 7.0.0 版及更新版本支援任務並行和佇列。
- Amazon EMR 7.3.0 版及更新版本預設會啟用任務並行和佇列。
- 您無法更新處於 STARTED 狀態的應用程式並行。
- 的有效範圍maxConcurrentRuns為 1 到 1000，而 queueTimeoutMinutes的有效範圍為 15 到 720。
- 帳戶最多可有 2000 個任務處於 QUEUED 狀態。
- 並行和佇列適用於批次和串流任務。它不能用於互動式任務。如需詳細資訊，請參閱[透過 EMR Studio 使用 EMR Serverless 執行互動式工作負載](#)。

使用 EMR Serverless 將資料取得至 S3 Express One Zone

使用 Amazon EMR 7.2.0 及更高版本時，請在執行任務和工作負載時搭配 [Amazon S3 Express One Zone](#) 儲存類別使用 EMR Serverless，以提高效能。S3 Express One Zone 是一種高效能的單一區域 Amazon S3 儲存類別，可為對大多數延遲敏感的應用程式提供一致的單一位數毫秒資料存取。在發布時，S3 Express One Zone 提供 Amazon S3 中最低延遲和最高效能的雲端物件儲存。

先決條件

- S3 Express One Zone 許可 – 當 S3 Express One Zone 最初在 S3 物件PUT上執行 GET、LIST或 等動作時，儲存類別CreateSession會代表您呼叫。您的 IAM 政策必須允許 s3express:CreateSession 許可，S3A 連接器才能調用 CreateSession API。如需具有此許可的範例政策，請參閱 [開始使用 S3 Express One Zone](#)。
- S3A 連接器 – 若要設定 Spark 從使用 Amazon S3 S3 儲存貯體存取資料，請使用 Apache Hadoop 連接器 S3A。若要使用該連接器，請確保所有 S3 URI 均使用 s3a 結構描述。如果沒有，請變更您用於 s3和 s3n結構描述的檔案系統實作。

若要變更 s3 結構描述，請指定下列叢集組態：

```
[
  {
    "Classification": "core-site",
    "Properties": {
      "fs.s3.impl": "org.apache.hadoop.fs.s3a.S3AFileSystem",
      "fs.AbstractFileSystem.s3.impl": "org.apache.hadoop.fs.s3a.S3A"
    }
  }
]
```

若要變更 s3n 結構描述，請指定下列叢集組態：

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

```
}  
}  
]
```

開始使用 S3 Express One Zone

請依照下列步驟開始使用 S3 Express One Zone。

1. [建立 VPC 端點](#)。將端點 `com.amazonaws.us-west-2.s3express` 新增至 VPC 端點。
2. 遵循 [Amazon EMR Serverless 入門](#)，使用 Amazon EMR 發行標籤 7.2.0 或更高版本建立應用程式。
3. [將應用程式設定為](#) 使用新建立的 VPC 端點、私有子網路群組和安全群組。
4. 將 `CreateSession` 許可新增至您的任務執行角色。

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Resource": [  
        "*" ]  
      ],  
      "Action": [  
        "s3express:CreateSession" ]  
      ],  
      "Sid": "AllowS3EXPRESSCreatesession"  
    }  
  ]  
}
```

5. 執行您的任務。請注意，使用 S3A 配置來存取 S3 Express One Zone 儲存貯體。

```
aws emr-serverless start-job-run \  
--application-id <application-id> \  
--execution-role-arn <job-role-arn> \  
--name <job-run-name> \  
--job-driver '{
```

```
"sparkSubmit": {  
  
  "entryPoint": "s3a://<DOC-EXAMPLE-BUCKET>/scripts/wordcount.py",  
  "entryPointArguments": ["s3a://<DOC-EXAMPLE-BUCKET>/emr-serverless-spark/output"],  
  "sparkSubmitParameters": "--conf spark.executor.cores=4  
--conf spark.executor.memory=8g --conf spark.driver.cores=4  
--conf spark.driver.memory=8g --conf spark.executor.instances=2  
--conf spark.hadoop.fs.s3a.change.detection.mode=none  
--conf spark.hadoop.fs.s3a.endpoint.region={<AWS_REGION>}  
--conf spark.hadoop.fs.s3a.select.enabled=false  
--conf spark.sql.sources.fastS3PartitionDiscovery.enabled=false  
}'
```

執行任務

佈建應用程式後，請將任務提交至應用程式。本節說明如何使用 AWS CLI 來執行這些任務。本節也會識別 EMR Serverless 上可用的每種應用程式類型的預設值。

主題

- [作業執行狀態](#)
- [具有寬限期的 EMR Serverless 任務執行取消](#)
- [從 EMR Studio 主控台執行任務](#)
- [從執行任務 AWS CLI](#)
- [執行 IAM 政策](#)
- [使用隨機最佳化磁碟](#)
- [將無伺服器儲存用於 Amazon EMR Serverless](#)
- [處理持續串流資料的串流任務](#)
- [執行 EMR Serverless 任務時使用 Spark 組態](#)
- [執行 EMR Serverless 任務時使用 Hive 組態](#)
- [EMR Serverless 任務彈性](#)
- [EMR Serverless 的中繼存放區組態](#)
- [從 EMR Serverless 存取另一個 AWS 帳戶中的 S3 資料](#)
- [故障診斷 EMR Serverless 中的錯誤](#)
- [啟用任務層級成本分配](#)

作業執行狀態

當您將任務執行提交至 Amazon EMR Serverless 任務佇列時，任務執行會進入 SUBMITTED 狀態。任務狀態會從 傳遞 SUBMITTED 到 ，RUNNING 直到達到 FAILED、SUCCESS 或 CANCELLING 為止。

作業執行可能有以下狀態：

State	Description
已提交	當您將任務執行提交至 EMR Serverless 時的初始任務狀態。任務會等待為應用程式排

State	Description
	程。EMR Serverless 開始排定任務執行的優先順序和排程。
已排入佇列	當應用程式層級任務執行並行完全佔用時，任務執行會在此狀態下等待。如需佇列和並行的詳細資訊，請參閱 EMR Serverless 應用程式的任務並行和佇列 。
待定	排程器正在評估任務執行，以排定應用程式的執行優先順序和排程。
已排程	EMR Serverless 已排程應用程式的作業執行，並正在配置資源來執行作業。
執行中	EMR Serverless 已配置任務最初需要的資源，且任務正在應用程式中執行。在 Spark 應用程式中，這意味著 Spark 驅動程式進程處於 running 狀態。
失敗	EMR Serverless 無法將任務執行提交至應用程式，或未成功完成。如需此任務失敗的其他資訊 <code>StateDetails</code> ，請參閱。
Success	任務執行已成功完成。
取消	<code>CancelJobRun</code> API 已請求取消任務執行，或任務執行已逾時。EMR Serverless 正在嘗試取消應用程式中的任務並釋出資源。
已取消	任務執行已成功取消，且已釋出其使用的資源。

具有寬限期的 EMR Serverless 任務執行取消

在資料處理系統中，突然終止可能會導致資源浪費、不完整的操作和潛在的資料不一致。Amazon EMR Serverless 可讓您在取消任務執行時指定寬限期。此功能允許在任務終止之前，有時間正確清除和完成進行中的工作。

Note

Amazon EMR 7.9.0 版及更新版本支援此功能。

取消任務執行時，請使用 參數指定寬限期（以秒為單位）`shutdownGracePeriodInSeconds`，在此期間任務可以在最終終止之前執行清除操作。行為和預設設定因批次和串流任務而異。

批次任務的寬限期

對於批次任務，EMR Serverless 可讓您實作在寬限期內執行的自訂清除操作。您可以在應用程式程式碼中將這些清除操作註冊為 JVM 關機掛鉤的一部分。

預設行為

關機的預設行為是沒有寬限期。它包含下列兩個動作：

- 立即終止
- 資源會立即釋出

組態選項

您可以指定導致正常關機的設定：

- 關閉寬限期的有效範圍：15-1800 秒（選用）
- 立即終止（無任何寬限期）：0 秒

啟用正常關機

若要實作批次任務的正常關閉，請遵循下列步驟：

1. 在包含自訂關閉邏輯的應用程式程式碼中新增關機掛鉤。

Example in Scala

```
import org.apache.hadoop.util.ShutdownHookManager

// Register shutdown hook with priority (second argument)
// Higher priority hooks run first
ShutdownHookManager.get().addShutdownHook(() => {
```

```
logger.info("Performing cleanup operations...")
}, 100)
```

使用 [ShutdownHookManager](#)

Example in PySpark

```
import atexit

def cleanup():
    # Your cleanup logic here
    print("Performing cleanup operations...")

# Register the cleanup function
atexit.register(cleanup)
```

2. 在取消任務時指定寬限期，以允許先前新增的勾點有時間執行

範例

```
# Default (immediate termination)
aws emr-serverless cancel-job-run \
  --application-id APPLICATION_ID \
  --job-run-id JOB_RUN_ID

# With 5-minute grace period
aws emr-serverless cancel-job-run \
  --application-id APPLICATION_ID \
  --job-run-id JOB_RUN_ID \
  --shutdown-grace-period-in-seconds 300
```

串流任務的寬限期

在 Spark 結構化串流中，其中運算涉及從外部資料來源讀取或寫入，突然關機可能會導致不需要的結果。串流任務處理微批次中的資料，並在中途中斷這些操作可能會導致後續嘗試中重複處理。當先前微批次的最新檢查點未寫入時，就會發生這種情況，導致串流任務重新啟動時再次處理相同的資料。這種重複處理不僅浪費運算資源，還可能影響業務營運，因此避免突然關機至關重要。

EMR Serverless 透過串流查詢接聽程式提供內建的正常關機支援。這可確保在任務終止之前正確完成持續的微批次。服務會自動管理串流應用程式微批次之間的正常關閉，確保目前的微批次完成處理、正確寫入檢查點，以及串流內容在關閉程序期間無擷取新資料的情況下，以乾淨的方式終止。

預設行為

- 預設啟用 120 秒寬限期。
- 內建串流查詢接聽程式可管理正常關機。

組態選項

- 關閉寬限期的有效範圍：15-1800 秒（選用）
- 立即終止：0 秒

啟用穩定關機

若要為串流任務實作正常關機：

取消任務時，請指定寬限期，讓進行中的微批次有時間完成。

範例

```
# Default graceful shutdown (120 seconds)
aws emr-serverless cancel-job-run \
  --application-id APPLICATION_ID \
  --job-run-id JOB_RUN_ID

# Custom grace period (e.g. 300 seconds)
aws emr-serverless cancel-job-run \
  --application-id APPLICATION_ID \
  --job-run-id JOB_RUN_ID \
  --shutdown-grace-period-in-seconds 300

# Immediate Termination
aws emr-serverless cancel-job-run \
  --application-id APPLICATION_ID \
  --job-run-id JOB_RUN_ID \
  --shutdown-grace-period-in-seconds 0
```

新增自訂關閉掛鉤（選用）

雖然 EMR Serverless 預設會透過內建串流查詢接聽程式來管理正常關機，但您可以選擇為個別串流查詢實作自訂關機邏輯。EMR Serverless 會註冊其具有優先順序 60 的正常關機接聽程式（使用

ShutdownHookManager)。由於優先順序較高的掛鉤會先執行，因此您可以註冊優先順序大於 60 的自訂清除操作，以確保它們在 EMR Serverless 的關閉程序開始之前執行。

若要新增自訂掛鉤，請參閱本主題中第一個範例，說明如何在應用程式程式碼中新增關機掛鉤。在這裡，100 是優先順序，大於 60。因此，這類關機掛鉤會先執行。

Note

自訂關閉掛鉤是選用的，對於正常的關閉功能並非必要，此功能由 EMR Serverless 自動處理。

寬限期費用和批次持續時間

如果使用寬限期的預設值 (120 秒)：

- 如果您的批次持續時間少於 120 秒，您只需支付完成批次所需的實際時間費用。
- 如果您的批次持續時間超過 120 秒，則會向您收取最長寬限期 (120 秒) 的費用，但查詢可能無法正常關閉，因為它將被強制終止。

若要最佳化成本並確保正常關機：

- 對於批次持續時間 > 120 秒：考慮增加寬限期以符合批次持續時間
- 對於 < 120 秒的批次持續時間：不需要調整寬限期，因為您只需支付實際處理時間的費用

考量事項

寬限期行為

- 寬限期提供註冊的關機掛鉤完成的時間。
- 任務會在關閉勾點完成後立即終止，即使它在寬限期之前也是如此。
- 如果清除操作超過寬限期，任務將強制終止。

服務行為

- 寬限期關閉僅適用於處於 RUNNING 狀態的任務。
- CANCELLING 狀態期間的後續取消請求會被忽略。

- 如果 EMR Serverless 因內部服務錯誤而無法啟動寬限期關閉：
 - 服務將重試最多 2 分鐘。
 - 如果重試失敗，任務將強制終止。

帳單

任務會針對使用的運算資源計費，直到任務完全關閉為止，包括在寬限期內採取的任何時間。

從 EMR Studio 主控台執行任務

您可以將任務執行提交至 EMR Serverless 應用程式，並從 EMR Studio 主控台存取任務。若要在 EMR Studio 主控台上建立或導覽至 EMR Serverless 應用程式，請遵循[從主控台入門](#)中的指示。

提交工作

在提交任務頁面上，將任務提交至 EMR Serverless 應用程式，如下所示。

Spark

1. 在名稱欄位中，輸入任務執行的名稱。
2. 在執行期角色欄位中，輸入 EMR Serverless 應用程式可以為任務執行擔任的 IAM 角色名稱。若要進一步了解執行期角色，請參閱 [Amazon EMR Serverless 的任務執行期角色](#)。
3. 在指令碼位置欄位中，輸入您要執行的指令碼或 JAR 的 Amazon S3 位置。對於 Spark 任務，指令碼可以是 Python (.py) 檔案或 JAR (.jar) 檔案。
4. 如果您的指令碼位置是 JAR 檔案，請在主類別欄位中輸入任務進入點的類別名稱。
5. (選用) 輸入其餘欄位的值。
 - 指令碼引數 — 輸入您要傳遞給主要 JAR 或 Python 指令碼的任何引數。您的程式碼會讀取這些參數。以逗號分隔陣列中的每個引數。
 - Spark 屬性 — 展開 Spark 屬性區段，並在此欄位中輸入任何 Spark 組態參數。

Note

如果您指定 Spark 驅動程式和執行器大小，請將記憶體負荷納入考量。在屬性 `spark.driver.memoryOverhead` 和 `spark.executor.memoryOverhead` 中指定記憶體額外負荷值。記憶體額外負荷的預設值為容器記憶體

的 10%，下限為 384 MB。執行器記憶體和記憶體額外負荷不能超過工作者記憶體。例如，30 GB 工作者 `spark.executor.memory` 的上限為 27 GB。

- 任務組態 — 在此欄位中指定任何任務組態。您可以使用這些任務組態來覆寫應用程式的預設組態。下列範例顯示如何覆寫 Spark 預設設定，例如執行器和驅動程式記憶體。

```
{
  "applicationConfiguration": [
    {
      "classification": "spark-defaults",
      "configurations": [],
      "properties": {
        "spark.executor.memory": "8G",
        "spark.driver.memory": "6G",
        "spark.driver.cores": "2",
        "spark.executor.cores": "4"
      }
    }
  ]
}
```

- 其他設定 — 啟用或停用 AWS Glue Data Catalog 做為中繼存放區，並修改應用程式日誌設定。若要進一步了解中繼存放區組態，請參閱 [EMR Serverless 的中繼存放區組態](#)。若要進一步了解應用程式記錄選項，請參閱 [儲存日誌](#)。
- 標籤 — 將自訂標籤指派給應用程式。

6. 選擇 Submit job (提交任務)。

Hive

1. 在名稱欄位中，輸入任務執行的名稱。
2. 在執行期角色欄位中，輸入 EMR Serverless 應用程式可為任務執行擔任的 IAM 角色名稱。
3. 在指令碼位置欄位中，輸入您要執行的指令碼或 JAR 的 Amazon S3 位置。對於 Hive 任務，指令碼必須是 Hive (.sql) 檔案。
4. (選用) 輸入其餘欄位的值。
 - 初始化指令碼位置 – 輸入在 Hive 指令碼執行之前初始化資料表的指令碼位置。
 - Hive 屬性 – 展開 Hive 屬性區段，並在此欄位中輸入任何 Hive 組態參數。

- 任務組態 – 指定任何任務組態。您可以使用這些任務組態來覆寫應用程式的預設組態。對於 Hive 任務，`hive.exec.scratchdir`和`hive.metastore.warehouse.dir`是hive-site組態中的必要屬性。

```
{
  "applicationConfiguration": [
    {
      "classification": "hive-site",
      "configurations": [],
      "properties": {
        "hive.exec.scratchdir": "s3://DOC-EXAMPLE_BUCKET/hive/scratch",
        "hive.metastore.warehouse.dir": "s3://DOC-EXAMPLE_BUCKET/hive/warehouse"
      }
    }
  ],
  "monitoringConfiguration": {}
}
```

- 其他設定 — 啟用或停用 AWS Glue Data Catalog 做為中繼存放區，並修改應用程式日誌設定。若要進一步了解中繼存放區組態，請參閱 [EMR Serverless 的中繼存放區組態](#)。若要進一步了解應用程式記錄選項，請參閱 [儲存日誌](#)。
- 標籤 — 將任何自訂標籤指派給應用程式。

5. 選擇 Submit job (提交任務)。

存取任務執行

從應用程式詳細資訊頁面上的任務執行索引標籤中，存取任務執行，並針對任務執行執行下列動作。

取消任務 — 若要取消處於 RUNNING 狀態的任務執行，請選擇此選項。若要進一步了解任務執行轉換，請參閱 [作業執行狀態](#)。

複製任務 — 若要複製先前的任務執行並重新提交，請選擇此選項。

從 執行任務 AWS CLI

您可以在上建立、描述和刪除個別任務 AWS CLI。您也可以列出所有任務，讓您一目了然地存取它們。

若要提交新任務，請使用 `start-job-run`。提供您要執行的應用程式 ID，以及任務特定的屬性。如需 Spark 範例，請參閱 [執行 EMR Serverless 任務時使用 Spark 組態](#)。如需 Hive 範例，請參閱 [執行 EMR Serverless 任務時使用 Hive 組態](#)。此命令會傳回您的 `application-id`、ARN 和新的 `job-id`。

每個任務執行都有設定的逾時持續時間。如果任務執行超過此持續時間，EMR Serverless 會自動將其取消。預設逾時為 12 小時。當您開始任務執行時，請將此逾時設定設定為符合您任務需求的值。使用 `executionTimeoutMinutes` 屬性設定值。

```
aws emr-serverless start-job-run \  
  --application-id application-id \  
  --execution-role-arn job-role-arn \  
  --execution-timeout-minutes 15 \  
  --job-driver '{  
    "hive": {  
      "query": "s3://amzn-s3-demo-bucket/scripts/create_table.sql",  
      "parameters": "--hiveconf hive.exec.scratchdir=s3://amzn-s3-demo-bucket/  
hive/scratch --hiveconf hive.metastore.warehouse.dir=s3://amzn-s3-demo-bucket/hive/  
warehouse"  
    }  
  }' \  
  --configuration-overrides '{  
    "applicationConfiguration": [{  
      "classification": "hive-site",  
      "properties": {  
        "hive.client.cores": "2",  
        "hive.client.memory": "4GIB"  
      }  
    }  
  ]}'
```

若要描述任務，請使用 `get-job-run`。此命令會傳回任務特定的組態，以及新任務的設定容量。

```
aws emr-serverless get-job-run \  
  --job-run-id job-id \  
  --application-id application-id
```

若要列出您的任務，請使用 `list-job-runs`。此命令會傳回一組縮寫屬性，其中包含任務類型、狀態和其他高階屬性。如果您不想存取所有任務，請指定您要存取的任務數量上限，最多 50 個。下列範例指定您想要存取前兩個任務執行。

```
aws emr-serverless list-job-runs \  
--max-results 2 \  
--application-id application-id
```

若要取消任務，請使用 `cancel-job-run`。提供您要取消之任務 `job-id` 的 `application-id` 和。

```
aws emr-serverless cancel-job-run \  
--job-run-id job-id \  
--application-id application-id
```

如需如何從 執行任務的詳細資訊 AWS CLI，請參閱 [EMR Serverless API 參考](#)。

執行 IAM 政策

在 EMR Serverless 上提交任務執行時，除了執行角色之外，您還可以指定執行 IAM 政策。任務執行所取得的許可是執行角色和指定執行 IAM 政策中許可的交集。

開始使用

使用執行 IAM 政策的步驟：

建立應用程式或使用現有的 `emr-serverless` 應用程式，然後執行下列 `aws cli`，以使用內嵌 IAM 政策開始任務執行：

```
aws emr-serverless start-job-run --region us-west-2 \  
--application-id application-id \  
--execution-role-arn execution-role-arn \  
--job-driver job-driver-options \  
--execution-iam-policy '{"policy": "inline-policy"}'
```

CLI 命令範例

如果我們在機器的 `policy.json` 檔案中存放了下列政策：

JSON

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

```

```

    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::my-test-bucket",
        "arn:aws:s3:::my-test-bucket/*"
      ],
      "Sid": "AllowS3GetObject"
    }
  ]
}

```

然後，我們可以使用下列 AWS CLI 命令啟動此政策的任務：

```

aws emr-serverless start-job-run --region us-west-2 \
  --application-id application-id \
  --execution-role-arn execution-role-arn \
  --job-driver job-driver-options \
  --execution-iam-policy '{
    "policy": '$(jq -c '. | @json' policy.json)'
  }'

```

您也可以同時使用 AWS 和客戶受管政策，透過其 ARNs 指定它們：

```

aws emr-serverless start-job-run --region us-west-2 \
  --application-id application-id \
  --execution-role-arn execution-role-arn \
  --job-driver job-driver-options \
  --execution-iam-policy '{
    "policyArns": [
      "arn:aws:iam::aws:policy/AmazonS3FullAccess",
      "arn:aws:iam::aws:policy/CloudWatchLogsFullAccess"
    ]
  }'

```

您也可以相同的請求中同時指定內嵌 IAM 政策和受管政策 ARNs：

```

aws emr-serverless start-job-run --region us-west-2 \

```

```
--application-id application-id \
--execution-role-arn execution-role-arn \
--job-driver job-driver-options
--execution-iam-policy '{
  "policy": '$(jq -c '. | @json' policy.json)',
  "policyArns": [
    "arn:aws:iam::aws:policy/AmazonS3FullAccess",
    "arn:aws:iam::aws:policy/CloudWatchLogsFullAccess"
  ]
}'
```

重要說明

- `execution-role-policy` 的欄位長度上限為 2048 個字元。
- 在的 `execution-iam-policy` 欄位中指定的內嵌 IAM 政策字串必須符合 json 字串標準，而不會逸出新行和引號，如上述範例所示。
- 最多可將 10 個受管政策 ARNs 的清單指定為 `execution-iam-policy` 的 `policyArns` 欄位的值。
- 受管政策 ARNs 必須是有效 AWS 或客戶受管政策 ARN 的清單，指定客戶受管政策 ARN 時，政策必須屬於 EMR-S JobRun 的相同 AWS 帳戶。
- 同時使用內嵌 IAM 政策和受管政策時，您用於內嵌和受管政策的純文字組合不得超過 2,048 個字元。
- JobRun 所取得的許可是執行角色和指定執行 IAM 政策中許可的交集。

政策交集

任務執行所取得的許可是執行角色和指定執行 IAM 政策中許可的交集。這表示必須在這兩個位置指定任何必要的許可，JobRun 才能運作。不過，您可以針對您不打算更新或覆寫的任何許可，在內嵌政策中指定額外的括號允許陳述式。

範例

根據下列執行 IAM 角色政策：

JSON

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

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "s3:*"
    ],
    "Resource": [
      "*"
    ],
    "Sid": "AllowS3"
  },
  {
    "Effect": "Allow",
    "Action": [
      "logs:DescribeLogGroups"
    ],
    "Resource": [
      "arn:aws:logs:us-west-2:123456789012:log-group:log-stream"
    ],
    "Sid": "AllowLOGSDescribeloggroups"
  },
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:DescribeTable"
    ],
    "Resource": [
      "arn:aws:dynamodb:*:*:table/MyCompany1table"
    ],
    "Sid": "AllowDYNAMOBDdescribetable"
  }
]
}

```

以及下列內嵌 IAM 政策：

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {

```

```

    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:ListBucket"
    ],
    "Resource": [
      "arn:aws:s3:::my-test-bucket/tenant1",
      "arn:aws:s3:::my-test-bucket/tenant1/*"
    ],
    "Sid": "AllowS3GetObject"
  },
  {
    "Effect": "Allow",
    "Action": [
      "logs:*",
      "dynamodb:*"
    ],
    "Resource": [
      "*"
    ],
    "Sid": "AllowLOGS"
  }
]
}

```

JobRun 所取得的許可如下：

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::my-test-bucket/tenant1",
        "arn:aws:s3:::my-test-bucket/tenant1/*"
      ],
    }
  ]
}

```

```
    "Sid": "AllowS3GetObject"
  },
  {
    "Effect": "Allow",
    "Action": [
      "logs:DescribeLogGroups"
    ],
    "Resource": [
      "arn:aws:logs:us-west-2:123456789012:log-group::log-stream"
    ],
    "Sid": "AllowLOGSDescribeLogGroups"
  },
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:DescribeTable"
    ],
    "Resource": [
      "arn:aws:dynamodb:*:*:table/MyCompany1table"
    ],
    "Sid": "AllowDYNAMODBDescribeTable"
  }
]
```

使用隨機最佳化磁碟

使用 Amazon EMR 7.1.0 版及更新版本，請在執行 Apache Spark 或 Hive 任務時使用隨機最佳化磁碟，以提高 I/O 密集型工作負載的效能。相較於標準磁碟，隨機播放最佳化磁碟提供更高的 IOPS（每秒 I/O 操作），可在隨機播放操作期間更快速地移動資料並減少延遲。隨機最佳化磁碟可讓您連接每個工作者最多 2 TB 的磁碟大小，因此請為您的工作負載需求設定適當的容量。

主要優點

隨機最佳化磁碟提供下列優點。

- 高 IOPS 效能 – 隨機播放最佳化磁碟提供的 IOPS 高於標準磁碟，因此在 Spark 和 Hive 任務和其他隨機播放密集型工作負載期間，資料隨機播放更有效率且速度更快。
- 較大的磁碟大小 – 隨機最佳化磁碟支援每個工作者 20GB 到 2TB 的磁碟大小，因此請根據您的工作負載選擇適當的容量。

開始使用

請參閱下列步驟，以在工作流程中使用隨機最佳化磁碟。

Spark

1. 使用下列命令建立 EMR Serverless 7.1.0 版應用程式。

```
aws emr-serverless create-application \  
  --type "SPARK" \  
  --name my-application-name \  
  --release-label emr-7.1.0 \  
  --region <AWS_REGION>
```

2. 設定您的 Spark 任務以包含參數 `spark.emr-serverless.driver.disk.type` 和 `spark.emr-serverless.executor.disk.type` 或使用隨機最佳化磁碟執行。視您的使用案例而定，您可以使用一個或兩個參數。

```
aws emr-serverless start-job-run \  
  --application-id application-id \  
  --execution-role-arn job-role-arn \  
  --job-driver '{  
    "sparkSubmit": {  
      "entryPoint": "/usr/lib/spark/examples/jars/spark-examples.jar",  
      "entryPointArguments": ["1"],  
      "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi  
      --conf spark.executor.cores=4  
      --conf spark.executor.memory=20g  
      --conf spark.driver.cores=4  
      --conf spark.driver.memory=8g  
      --conf spark.executor.instances=1  
      --conf spark.emr-serverless.executor.disk.type=shuffle_optimized"  
    }  
  }'
```

如需詳細資訊，請參閱 [Spark 任務屬性](#)。

Hive

1. 使用下列命令建立 EMR Serverless 7.1.0 版應用程式。

```
aws emr-serverless create-application \  
  --type "HIVE" \  
  --name my-application-name \  
  --release-label emr-7.1.0 \  
  --region <AWS_REGION>
```

2. 設定您的 Hive 任務以包含參數 `hive.driver.disk.type` 和 `hive.tez.disk.type` 或使用隨機最佳化磁碟執行。視您的使用案例而定，您可以使用一個或兩個參數。

```
aws emr-serverless start-job-run \  
  --application-id application-id \  
  --execution-role-arn job-role-arn \  
  --job-driver '{  
    "hive": {  
      "query": "s3://<DOC-EXAMPLE-BUCKET>/emr-serverless-hive/query/hive-  
query.ql",  
      "parameters": "--hiveconf hive.log.explain.output=false"  
    }  
  }' \  
  --configuration-overrides '{  
    "applicationConfiguration": [{  
      "classification": "hive-site",  
      "properties": {  
        "hive.exec.scratchdir": "s3://<DOC-EXAMPLE-BUCKET>/emr-  
serverless-hive/hive/scratch",  
        "hive.metastore.warehouse.dir": "s3://<DOC-EXAMPLE-BUCKET>/emr-  
serverless-hive/hive/warehouse",  
        "hive.driver.cores": "2",  
        "hive.driver.memory": "4g",  
        "hive.tez.container.size": "4096",  
        "hive.tez.cpu.vcores": "1",  
        "hive.driver.disk.type": "shuffle_optimized",  
        "hive.tez.disk.type": "shuffle_optimized"  
      }  
    }  
  ]  
}'
```

如需詳細資訊，請參閱 [Hive 任務屬性](#)。

設定具有預先初始化容量的應用程式

請參閱下列範例，以根據 Amazon EMR 7.1.0 版建立應用程式。這些應用程式具有下列屬性：

- 5 個預先初始化的 Spark 驅動程式，每個驅動程式都有 2 個 vCPU、4 GB 記憶體和 50 GB 隨機最佳化磁碟。
- 50 個預先初始化的執行器，每個執行器都有 4 個 vCPU、8 GB 記憶體和 500 GB 隨機最佳化磁碟。

當此應用程式執行 Spark 任務時，它會先使用預先初始化的工作者，然後將隨需工作者擴展到最大容量 400 個 vCPU 和 1024 GB 的記憶體。或者，您可以省略 DRIVER 或 的容量 EXECUTOR。

Spark

```
aws emr-serverless create-application \  
  --type "SPARK" \  
  --name <my-application-name> \  
  --release-label emr-7.1.0 \  
  --initial-capacity '{  
    "DRIVER": {  
      "workerCount": 5,  
      "workerConfiguration": {  
        "cpu": "2vCPU",  
        "memory": "4GB",  
        "disk": "50GB",  
        "diskType": "SHUFFLE_OPTIMIZED"  
      }  
    },  
    "EXECUTOR": {  
      "workerCount": 50,  
      "workerConfiguration": {  
        "cpu": "4vCPU",  
        "memory": "8GB",  
        "disk": "500GB",  
        "diskType": "SHUFFLE_OPTIMIZED"  
      }  
    }  
  }' \  
  --maximum-capacity '{  
    "cpu": "400vCPU",  
    "memory": "1024GB"  
  }'
```

Hive

```
aws emr-serverless create-application \  
--type "HIVE" \  
--name <my-application-name> \  
--release-label emr-7.1.0 \  
--initial-capacity '{  
  "DRIVER": {  
    "workerCount": 5,  
    "workerConfiguration": {  
      "cpu": "2vCPU",  
      "memory": "4GB",  
      "disk": "50GB",  
      "diskType": "SHUFFLE_OPTIMIZED"  
    }  
  },  
  "EXECUTOR": {  
    "workerCount": 50,  
    "workerConfiguration": {  
      "cpu": "4vCPU",  
      "memory": "8GB",  
      "disk": "500GB",  
      "diskType": "SHUFFLE_OPTIMIZED"  
    }  
  }  
}' \  
--maximum-capacity '{  
  "cpu": "400vCPU",  
  "memory": "1024GB"  
}'
```

將無伺服器儲存用於 Amazon EMR Serverless

使用 Amazon EMR 7.12 版及更高版本時，請在執行 Apache Spark 任務時使用無伺服器儲存體，以消除本機磁碟佈建並降低資料處理成本，並防止任務失敗造成磁碟容量限制。無伺服器儲存會自動處理任務的隨機播放、磁碟溢出和磁碟快取操作，而不需要容量組態，並免費存放中繼資料。Amazon EMR Serverless 會將中繼資料存放在全受管無伺服器儲存體中，該儲存體會根據工作負載需求自動擴展，並可讓 Spark 在閒置時立即釋放運算工作者，進而降低運算成本。

主要優點

EMR Serverless 的無伺服器儲存提供下列優點。

- 零組態儲存 – 無伺服器儲存不需要為每個應用程式或任務設定本機磁碟類型和大小。EMR Serverless 會自動管理中繼資料操作，無需規劃容量。
- 透過自動擴展來防止任務失敗 – 儲存容量會根據工作負載需求自動擴展，防止任務失敗導致磁碟容量不足。
- 降低資料處理成本 – 無伺服器儲存體透過兩種機制降低處理成本。首先，免費提供中繼資料儲存，您只需支付運算和記憶體資源的費用。其次，具有 Spark 動態資源配置的解耦儲存可讓 Spark 在閒置時立即釋放工作者，而不是保留工作者，以保留本機磁碟上的中繼資料。這可加快每個 Spark 階段的向外擴展和向內擴展速度，降低較新階段需要較少工作者的任務運算成本。
- 具有任務層級隔離的加密儲存 – 所有中繼資料都會在傳輸中和靜態時以嚴格的任務層級隔離進行加密。
- 精細存取控制支援 – 無伺服器儲存支援透過 AWS Lake Formation 整合進行精細存取控制。

開始使用

請參閱下列步驟，以在 Spark 工作流程中使用 EMR Serverless 的無伺服器儲存。

1. 建立 EMR Serverless 應用程式

在 spark-defaults 分類中將 spark 屬性設定為 `spark.aws.serverlessStorage.enabled true`，以建立已啟用無伺服器儲存的 EMR Serverless 7.12 版（或更新版本）應用程式。

```
aws emr-serverless create-application \  
  --type "SPARK" \  
  --name my-application \  
  --release-label emr-7.12.0 \  
  --runtime-configuration '[{  
    "classification": "spark-defaults",  
    "properties": {  
      "spark.aws.serverlessStorage.enabled": "true"  
    }  
  }]' \  
  --region <AWS_REGION>
```

2. 啟動 Spark 任務

在您的應用程式上啟動任務執行。EMR Serverless 的無伺服器儲存會自動處理中繼資料操作，例如任務的隨機播放。

```
aws emr-serverless start-job-run \  
  --application-id <application-id> \  
  --execution-role-arn <job-role-arn> \  
  --job-driver '{  
    "sparkSubmit": {  
      "entryPoint": "s3://<bucket>/script.py",  
      "sparkSubmitParameters": "--conf spark.executor.cores=4  
        --conf spark.executor.memory=20g  
        --conf spark.driver.cores=4  
        --conf spark.driver.memory=8g  
        --conf spark.executor.instances=10"  
    }  
  }'
```

您也可以任務層級為 EMR Serverless 啟用無伺服器儲存，即使應用程式層級未啟用也一樣。這將啟動已啟用無伺服器儲存的工作者節點，以處理您的任務。您也可以將相同的 Spark 屬性設定為 `spark.aws.serverlessStorage.enabled false`，以停用特定任務的無伺服器儲存。

```
# Turn on serverless storage for EMR serverless for a specific job  
aws emr-serverless start-job-run \  
  --application-id <application-id> \  
  --execution-role-arn <job-role-arn> \  
  --job-driver '{  
"sparkSubmit": {  
"entryPoint": "/usr/lib/spark/examples/jars/spark-examples.jar",  
  "entryPointArguments": ["1"],  
  "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi  
    --conf spark.aws.serverlessStorage.enabled": "true"  
  }  
}'
```

Note

若要繼續使用傳統的本機磁碟佈建，請省略 `spark.aws.serverlessStorage.enabled` 組態或將其設定為 `false`。

考量和限制

- 發行版本 – Amazon EMR 7.12 版及更新版本支援無伺服器儲存。
- 資料量限制 – 每個任務每次任務執行最多可讀取和寫入總計 200 GB 的中繼資料。超過此限制的任務將會失敗，並顯示錯誤訊息，指出已達到無伺服器儲存限制。
- 任務執行逾時 – 無伺服器儲存支援執行逾時長達 24 小時的任務。針對較長執行逾時設定的任務將會失敗，並顯示錯誤訊息。
- 預先初始化的容量 – 預先初始化的容量工作者不支援無伺服器儲存。當您設定預先初始化的容量時，它只會由在任務層級明確停用無伺服器儲存的任務使用。啟用無伺服器儲存的任務一律會隨需佈建新工作者，而且不會使用任何預先初始化的容量，無論應用程式層級中的組態為何。
- 工作負載類型 – 串流和互動式任務不支援無伺服器儲存。
- 工作者組態 – 具有 1 或 2 個 vCPUs 工作者不支援無伺服器儲存。

支援的 AWS 區域

EMR Serverless 在下列區域中支援無伺服器儲存：

- 美國東部 (維吉尼亞北部)
- 美國東部 (俄亥俄)
- 美國西部 (加利佛尼亞北部)
- 美國西部 (奧勒岡)
- 非洲 (開普敦)
- 亞太地區 (香港)
- 亞太地區 (雅加達)
- 亞太地區 (墨爾本)
- 亞太地區 (孟買)
- 亞太區域 (大阪)
- 亞太區域 (首爾)
- 亞太區域 (新加坡)
- 亞太地區 (雪梨)
- 亞太區域 (東京)
- 加拿大 (中部)
- 加拿大西部 (卡加利)

- 歐洲 (法蘭克福)
- 歐洲 (愛爾蘭)
- 歐洲 (倫敦)
- 歐洲 (米蘭)
- Europe (Paris)
- 歐洲 (西班牙)
- 歐洲 (斯德哥爾摩)
- 歐洲 (蘇黎世)
- 南美洲 (聖保羅)

處理持續串流資料的串流任務

EMR Serverless 中的串流任務是一種任務模式，可讓您近乎即時地分析和處理串流資料。這些長時間執行的任務會輪詢串流資料，並在資料送達時持續處理結果。串流任務最適合需要即時資料處理的任務，例如近乎即時的分析、詐騙偵測和建議引擎。EMR Serverless 串流任務提供最佳化，例如內建任務彈性、即時監控、增強型日誌管理，以及與串流連接器的整合。

以下是串流任務的一些使用案例：

- 近乎即時的分析 – Amazon EMR Serverless 中的串流任務可讓您近乎即時地處理串流資料，因此您可以對連續資料串流執行即時分析，例如日誌資料、感應器資料或點擊串流資料，以衍生洞見並根據最新資訊及時做出決策。
- 詐騙偵測 – 當您分析資料串流並識別可疑模式或異常時，使用串流任務在金融交易、信用卡操作或線上活動中執行近乎即時的詐騙偵測。
- 建議引擎 – 串流任務可以處理使用者活動資料和更新建議模型。這樣做會根據行為和偏好開啟個人化和即時建議的可能性。
- 社交媒體分析 – 串流任務可以處理社交媒體資料，例如推文、評論和文章，讓組織可以近乎即時地監控趨勢、情緒分析和品牌評價。
- 物聯網 (IoT) 分析 – 串流任務可以處理和分析來自 IoT 裝置、感應器和連線機器的高速資料串流，因此請執行異常偵測、預測性維護和其他 IoT 分析使用案例。
- Clickstream 分析 – 串流任務可以處理和分析來自網站或行動應用程式的 Clickstream 資料。使用這類資料的企業可以執行分析，以進一步了解使用者行為、個人化使用者體驗，以及最佳化行銷活動。
- 日誌監控和分析 – 串流任務也可以處理來自伺服器、應用程式和網路裝置的日誌資料。這可為您提供異常偵測、故障診斷，以及系統運作狀態和效能。

主要優點

EMR Serverless 中的串流任務會自動提供任務彈性，這是下列因素的組合：

- 自動重試 – EMR Serverless 會自動重試失敗的任何任務，而無需您進行任何手動輸入。
- 可用區域 (AZ) 彈性 – 如果原始可用區域遇到問題，EMR Serverless 會自動將串流任務切換到運作狀態良好的可用區域。
- 日誌管理：
 - 日誌輪換 – 為了更有效率的磁碟儲存管理，EMR Serverless 會定期輪換長時間串流任務的日誌。這樣做可以防止可能耗用所有磁碟空間的日誌累積。
 - 日誌壓縮 – 可協助您在受管持久性中有效率地管理和最佳化日誌檔案。壓縮也會改善您使用受管 Spark 歷史記錄伺服器的偵錯體驗。

支援的資料來源和資料接收器

EMR Serverless 可與多個輸入資料來源和輸出資料接收器搭配使用：

- 支援的輸入資料來源 – Amazon Kinesis Data Streams、Amazon Managed Streaming for Apache Kafka 和自我管理 Apache Kafka 叢集。根據預設，Amazon EMR 7.1.0 版和更新版本包含 [Amazon Kinesis Data Streams 連接器](#)，因此您不需要建置或下載任何其他套件。
- 支援的輸出資料接收器 – AWS Glue Data Catalog 資料表、Amazon S3、Amazon Redshift、MySQL、PostgreSQL Oracle、Oracle、Microsoft SQL、Apache Iceberg、Delta Lake 和 Apache Hudi。

考量和限制

當您使用串流任務時，請記住下列考量和限制。

- [Amazon EMR 7.1.0 版及更高版本](#) 支援串流任務。
- EMR Serverless 預期串流任務會長時間執行，因此您無法設定執行逾時來限制任務的執行時間。
- 串流任務僅與 Spark 引擎相容，該引擎以 [結構化串流架構](#) 為基礎。
- EMR Serverless 會無限期重試串流任務，您無法自訂最大嘗試次數。如果失敗的嘗試次數超過每小時時段設定的閾值，則會自動包含防限流功能以停止任務重試。預設閾值是在一小時內嘗試失敗 5 次。您可以將此閾值設定為 1 到 10 次嘗試。如需詳細資訊，請參閱 [任務彈性](#)。
- 串流任務具有檢查點來儲存執行時間狀態和進度，因此 EMR Serverless 可以從最新的檢查點繼續串流任務。如需詳細資訊，請參閱 Apache Spark 文件中的 [使用檢查點從失敗中復原](#)。

開始使用串流任務

請參閱下列指示，了解如何開始使用串流任務。

1. 遵循 [Amazon EMR Serverless 入門來建立應用程式](#)。請注意，您的應用程式必須執行 [Amazon EMR 7.1.0 版](#)或更新版本。
2. 一旦應用程式準備就緒，請將 mode 參數設定為 STREAMING 以提交串流任務，類似下列 AWS CLI 範例。

```
aws emr-serverless start-job-run \  
--application-id <APPLICATION_ID> \  
--execution-role-arn <JOB_EXECUTION_ROLE> \  
--mode 'STREAMING' \  
--job-driver '{  
  "sparkSubmit": {  
    "entryPoint": "s3://<streaming script>",  
    "entryPointArguments": ["s3://<DOC-EXAMPLE-BUCKET-OUTPUT>/output"],  
    "sparkSubmitParameters": "--conf spark.executor.cores=4  
      --conf spark.executor.memory=16g  
      --conf spark.driver.cores=4  
      --conf spark.driver.memory=16g  
      --conf spark.executor.instances=3"  
  }  
'
```

支援的串流連接器

串流連接器有助於從串流來源讀取資料，也可以將資料寫入串流接收器。

以下是支援的串流連接器：

Amazon Kinesis Data Streams 連接器

適用於 Apache Spark 的 [Amazon Kinesis Data Streams 連接器](#)可建置串流應用程式和管道，以取用來自的資料，並將資料寫入 Amazon Kinesis Data Streams。連接器支援增強的廣發耗用，專用讀取輸送量速率最高可達每個碎片 2MB/秒。根據預設，Amazon EMR Serverless 7.1.0 及更高版本包含連接器，因此您不需要建置或下載任何其他套件。如需連接器的詳細資訊，請參閱 [GitHub 上的 spark-sql-kinesis-connector 頁面](#)。

以下是如何使用 Kinesis Data Streams 連接器相依性啟動任務執行的範例。

```
aws emr-serverless start-job-run \
--application-id <APPLICATION_ID> \
--execution-role-arn <JOB_EXECUTION_ROLE> \
--mode 'STREAMING' \
--job-driver '{
  "sparkSubmit": {
    "entryPoint": "s3://<Kinesis-streaming-script>",
    "entryPointArguments": ["s3://<DOC-EXAMPLE-BUCKET-OUTPUT>/output"],
    "sparkSubmitParameters": "--conf spark.executor.cores=4
      --conf spark.executor.memory=16g
      --conf spark.driver.cores=4
      --conf spark.driver.memory=16g
      --conf spark.executor.instances=3
      --jars /usr/share/aws/kinesis/spark-sql-kinesis/lib/spark-streaming-
sql-kinesis-connector.jar"
  }
}'
```

若要連線至 Kinesis Data Streams，請使用 VPC 存取設定 EMR Serverless 應用程式，並使用 VPC 端點來允許私有存取。或使用 NAT Gateway 來取得公有存取。如需詳細資訊，請參閱[設定 VPC 存取](#)。您還必須確保任務執行時間角色具有必要的讀取和寫入許可，以存取所需的資料串流。若要進一步了解如何設定任務執行期角色，請參閱[Amazon EMR Serverless 的任務執行期角色](#)。如需所有必要許可的完整清單，請參閱[GitHub 上的 spark-sql-kinesis-connector 頁面](#)。

Apache Kafka 連接器

適用於 Spark 結構化串流的 Apache Kafka 連接器是 Spark 社群的開放原始碼連接器，可在 Maven 儲存庫中使用。此連接器有助於 Spark 結構化串流應用程式從自我管理的 Apache Kafka 和 Amazon Managed Streaming for Apache Kafka 讀取和寫入資料。如需連接器的詳細資訊，請參閱 Apache Spark 文件中的[結構化串流 + Kafka 整合指南](#)。

下列範例示範如何在任務執行請求中包含 Kafka 連接器。

```
aws emr-serverless start-job-run \
--application-id <APPLICATION_ID> \
--execution-role-arn <JOB_EXECUTION_ROLE> \
--mode 'STREAMING' \
--job-driver '{
  "sparkSubmit": {
    "entryPoint": "s3://<Kafka-streaming-script>",
    "entryPointArguments": ["s3://<DOC-EXAMPLE-BUCKET-OUTPUT>/output"],
```

```

    "sparkSubmitParameters": "--conf spark.executor.cores=4
      --conf spark.executor.memory=16g
      --conf spark.driver.cores=4
      --conf spark.driver.memory=16g
      --conf spark.executor.instances=3
      --packages org.apache.spark:spark-sql-
kafka-0-10_2.12:<KAFKA_CONNECTOR_VERSION>"
  }
}'

```

Apache Kafka 連接器版本取決於您的 EMR Serverless 發行版本和對應的 Spark 版本。若要尋找正確的 Kafka 版本，請參閱 [結構化串流 + Kafka 整合指南](#)。

若要搭配 IAM 身分驗證使用 Amazon Managed Streaming for Apache Kafka，請包含另一個相依性，讓 Kafka 連接器能夠透過 IAM 連線至 Amazon MSK。如需詳細資訊，請參閱 [GitHub 上的 aws-msk-iam-auth 儲存庫](#)。您也必須確保任務執行時間角色具有必要的 IAM 許可。下列範例示範如何使用具有 IAM 身分驗證的連接器。

```

aws emr-serverless start-job-run \
--application-id <APPLICATION_ID> \
--execution-role-arn <JOB_EXECUTION_ROLE> \
--mode 'STREAMING' \
--job-driver '{
  "sparkSubmit": {
    "entryPoint": "s3://<Kafka-streaming-script>",
    "entryPointArguments": ["s3://<DOC-EXAMPLE-BUCKET-OUTPUT>/output"],
    "sparkSubmitParameters": "--conf spark.executor.cores=4
      --conf spark.executor.memory=16g
      --conf spark.driver.cores=4
      --conf spark.driver.memory=16g
      --conf spark.executor.instances=3
      --packages org.apache.spark:spark-sql-
kafka-0-10_2.12:<KAFKA_CONNECTOR_VERSION>,software.amazon.msk:aws-msk-iam-
auth:<MSK_IAM_LIB_VERSION>"
  }
}'

```

若要從 Amazon MSK 使用 Kafka 連接器和 IAM 身分驗證程式庫，請設定具有 VPC 存取的 EMR Serverless 應用程式。您的子網路必須具有網際網路存取權，並使用 NAT Gateway 來存取 Maven 相依性。如需詳細資訊，請參閱 [設定 VPC 存取](#)。子網路必須具有網路連線才能存取 Kafka 叢集。無論您的 Kafka 叢集是自我管理，或是您使用 Amazon Managed Streaming for Apache Kafka，都是如此。

串流任務日誌管理

串流任務支援 Spark 應用程式日誌和事件日誌的日誌輪換，以及 Spark 事件日誌的日誌壓縮。這可協助您有效管理資源。

日誌輪換

串流任務支援 Spark 應用程式日誌和事件日誌的日誌輪換。日誌輪換可防止長串流任務產生可能佔用所有可用磁碟空間的大型日誌檔案。日誌輪換可協助您節省磁碟儲存體，並防止任務因磁碟空間不足而失敗。如需詳細資訊，請參閱[輪換日誌](#)。

日誌壓縮

每當有受管記錄可用時，串流任務也支援 Spark 事件日誌的日誌壓縮。如需受管記錄的詳細資訊，請參閱[使用受管儲存記錄](#)。串流任務可以長時間執行，而且事件資料量會隨著時間累積，並大幅增加日誌檔案大小。Spark 歷史記錄伺服器會讀取這些事件並將其載入至 Spark 應用程式 UI 的記憶體。此程序可能會導致高延遲和成本，特別是如果存放在 Amazon S3 中的事件日誌非常大時。

日誌壓縮會減少事件日誌大小，因此 Spark 歷史記錄伺服器不需要隨時載入超過 1 GB 的事件日誌。如需詳細資訊，請參閱 Apache Spark 文件中的[監控和檢測](#)。

執行 EMR Serverless 任務時使用 Spark 組態

您可以在 type 參數設定為的應用程式上執行 Spark 任務 SPARK。任務必須與 Amazon EMR 發行版本相容的 Spark 版本相容。例如，當您使用 Amazon EMR 6.6.0 版執行任務時，您的任務必須與 Apache Spark 3.2.0 相容。如需每個版本的應用程式版本資訊，請參閱[Amazon EMR Serverless 發行版本](#)。

Spark 任務參數

當您使用 [StartJobRun API](#) 執行 Spark 任務時，請指定下列參數。

必要參數

- [Spark 任務執行期角色](#)
- [Spark 任務驅動程式參數](#)
- [Spark 組態覆寫參數](#)
- [Spark 動態資源配置最佳化](#)

Spark 任務執行期角色

使用 `executionRoleArn` 為應用程式用來執行 Spark 任務的 IAM 角色指定 ARN。此角色必須包含下列許可：

- 從資料所在的 S3 儲存貯體或其他資料來源讀取
- 從 PySpark 指令碼或 JAR 檔案所在的 S3 儲存貯體或字首讀取
- 寫入您要寫入最終輸出的 S3 儲存貯體
- 將日誌寫入 S3MonitoringConfiguration 指定的 S3 儲存貯體或字首
- 如果您使用 KMS 金鑰來加密 S3 儲存貯體中的資料，則存取 KMS 金鑰
- 如果您使用 SparkSQL，存取 AWS Glue Data Catalog

如果您的 Spark 任務讀取資料或從其他資料來源寫入資料，請在此 IAM 角色中指定適當的許可。如果您未將這些許可提供給 IAM 角色，任務可能會失敗。如需詳細資訊，請參閱 [Amazon EMR Serverless 的任務執行期角色](#) 和 [儲存日誌](#)。

Spark 任務驅動程式參數

使用 `jobDriver` 為任務提供輸入。任務驅動程式參數只接受您要執行之任務類型的一個值。對於 Spark 任務，參數值為 `sparkSubmit`。您可以使用此任務類型，透過 Spark 提交執行 Scala、Java、PySpark 和任何其他支援的任務。Spark 任務具有下列參數：

- **sparkSubmitParameters** – 這些是您要傳送至任務的其他 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執行器數量，例如 `--conf` 或 `--class` 引數中定義的屬性。
- **entryPointArguments** – 這是您要傳遞至主要 JAR 或 Python 檔案的引數陣列。應使用 `entrypoint` 程式碼讀取這些參數。以逗號分隔陣列中的每個引數。
- **entryPoint** – 這是 Amazon S3 中您要執行的主要 JAR 或 Python 檔案的參考。如果您正在執行 Scala 或 Java JAR，`sparkSubmitParameters` 請使用 `--class` 引數在 `entryPoint` 中指定主要項目類別。

如需詳細資訊，請參閱 [使用 spark-submit 啟動應用程式](#)。

Spark 組態覆寫參數

使用 `configurationOverrides` 覆寫監控層級和應用程式層級組態屬性。此參數接受具有下列兩個欄位的 JSON 物件：

- **monitoringConfiguration** - 使用此欄位指定您希望 EMR Serverless 任務存放 Spark 任務日誌的 Amazon S3 URL (s3MonitoringConfiguration)。請確定您已使用託管您應用程式的 AWS 帳戶相同，以及在執行任務 AWS 區域的相同中建立此儲存貯體。
- **applicationConfiguration** - 若要覆寫應用程式的預設組態，您可以在此欄位中提供組態物件。可以使用速記語法，以提供組態或參考 JSON 檔案中物件的組態。組態物件是由分類、屬性和選用的巢狀組態所組成。屬性由您想要在檔案中覆寫的設定組成。您可以在單一 JSON 物件中，為多個應用程式指定多個分類。

Note

可用的組態分類會因特定 EMR Serverless 版本而有所不同。例如，自訂 Log4j spark-driver-log4j2 和 的分類 spark-executor-log4j2 僅適用於 6.8.0 版和更新版本。

如果您在應用程式覆寫和 Spark 提交參數中使用相同的組態，Spark 提交參數會優先。組態的優先順序如下，從最高到最低：

- EMR Serverless 在建立時提供的組態 SparkSession。
- 您隨 sparkSubmitParameters 引 --conf 數一起提供的組態。
- 當您啟動任務時，您作為應用程式一部分提供的組態會覆寫。
- 您在建立應用程式 runtimeConfiguration 時作為的一部分提供的組態。
- Amazon EMR 用於發行版本的最佳化組態。
- 應用程式的預設開放原始碼組態。

如需在應用程式層級宣告組態，以及在任務執行期間覆寫組態的詳細資訊，請參閱 [EMR Serverless 的預設應用程式組態](#)。

Spark 動態資源配置最佳化

使用 dynamicAllocationOptimization 來最佳化 EMR Serverless 中的資源用量。在 Spark 組態分類 true 中將此屬性設定為表示為 EMR Serverless，以最佳化執行器資源配置，讓 Spark 請求和取消執行器的速率與 EMR Serverless 建立和釋放工作者的速率更一致。如此一來，EMR Serverless 就能在各個階段以最佳方式重複使用工作者，進而在執行具有多個階段的任務時降低成本，同時維持相同的效能。

此屬性適用於所有 Amazon EMR 發行版本。

以下是使用的範例組態分類dynamicAllocationOptimization。

```
[
  {
    "Classification": "spark",
    "Properties": {
      "dynamicAllocationOptimization": "true"
    }
  }
]
```

如果您使用的是動態配置最佳化，請考慮下列事項：

- 此最佳化適用於您啟用動態資源配置的 Spark 任務。
- 為了實現最佳成本效益，我們建議您根據您的工作負載，使用任務層級設定spark.dynamicAllocation.maxExecutors或[應用程式層級最大容量](#)設定來設定工作者上擴展範圍。
- 您可能不會注意到更簡單任務的成本改善。例如，如果您的任務在小型資料集上執行或完成在一個階段中執行，Spark 可能不需要更多執行器或多個擴展事件。
- 具有一系列大型階段、較小階段，然後再次大型階段的任務，可能會在任務執行時間中遇到迴歸。隨著 EMR Serverless 更有效率地使用資源，它可能會導致較大型階段的可用工作者減少，進而延長執行時間。

Spark 任務屬性

下表列出選用的 Spark 屬性及其預設值，您可以在提交 Spark 任務時覆寫這些屬性。

選用 Spark 屬性和預設值

金鑰	Description	預設值
spark.archives	Spark 擷取到每個執行器工作目錄的以逗號分隔的封存清單。支援的檔案類型包括 .jar、.tar.gz、.tgz和 .zip。若要指定要擷取的目錄名稱，請在您要擷取的檔案名稱#後	NULL

金鑰	Description	預設值
	面新增。例如 <code>file.zip#directory</code> 。	
<code>spark.authenticate</code>	開啟 Spark 內部連線身分驗證的選項。	TRUE
<code>spark.driver.cores</code>	驅動程式使用的核心數量。	4
<code>spark.driver.extraJavaOptions</code>	Spark 驅動程式的額外 Java 選項。	NULL
<code>spark.driver.memory</code>	驅動程式使用的記憶體量。	14G
<code>spark.dynamicAllocation.enabled</code>	開啟動態資源配置的選項。此選項會根據工作負載，縱向擴展或縮減向應用程式註冊的執行器數量。	TRUE
<code>spark.dynamicAllocation.executorIdleTimeout</code>	在 Spark 移除執行器之前，執行器可以保持閒置的時間長度。這僅適用於您開啟動態配置的情況。	60 年代
<code>spark.dynamicAllocation.initialExecutors</code>	如果您開啟動態配置，要執行的執行器初始數量。	3
<code>spark.dynamicAllocation.maxExecutors</code>	如果您開啟動態配置，執行器數量的上限。	對於 6.10.0 和更新版本， infinity 對於 6.9.0 及更低版本，100
<code>spark.dynamicAllocation.minExecutors</code>	如果您開啟動態配置，執行器數量的下限。	0
<code>spark.emr-serverless.allocation.batch.size</code>	每個執行器配置週期中要請求的容器數量。每個配置週期之間有一秒的間隔。	20

金鑰	Description	預設值
<code>spark.emr-serverless.driver.disk</code>	Spark 驅動程式磁碟。	20G
<code>spark.emr-serverless.driverEnv.</code> [KEY]	將環境變數新增至 Spark 驅動程式的選項。	NULL
<code>spark.emr-serverless.executor.disk</code>	Spark 執行器磁碟。	20G
<code>spark.emr-serverless.memoryOverheadFactor</code>	設定要新增至驅動程式和執行器容器記憶體之記憶體額外負荷。	0.1
<code>spark.emr-serverless.driver.disk.type</code>	連接至 Spark 驅動程式的磁碟類型。	標準
<code>spark.emr-serverless.executor.disk.type</code>	連接至 Spark 執行器的磁碟類型。	標準
<code>spark.executor.cores</code>	每個執行器使用的核心數量。	4
<code>spark.executor.extraJavaOptions</code>	Spark 執行器的額外 Java 選項。	NULL
<code>spark.executor.instances</code>	要配置的 Spark 執行器容器數量。	3
<code>spark.executor.memory</code>	每個執行器使用的記憶體數量。	14G
<code>spark.executorEnv.</code> [KEY]	將環境變數新增至 Spark 執行器的選項。	NULL

金鑰	Description	預設值
<code>spark.files</code>	要放入每個執行器工作目錄中的檔案逗號分隔清單。您可以使用存取執行器中這些檔案的檔案路徑 <code>SparkFile s.get(<i>fileName</i>)</code> 。	NULL
<code>spark.hadoop.hive.metastore.client.factory.class</code>	Hive 中繼存放區實作類別。	NULL
<code>spark.jars</code>	要新增至驅動程式和執行器執行時間 classpath 的其他 jar。	NULL
<code>spark.network.crypto.enabled</code>	開啟 AES 型 RPC 加密的選項。這包括在 Spark 2.2.0 中新增的身分驗證通訊協定。	FALSE
<code>spark.sql.warehouse.dir</code>	受管資料庫和資料表的預設位置。	的值 <code>\$PWD/spark-warehouse</code>
<code>spark.submit.pyFiles</code>	以逗號分隔的 <code>.zip</code> 、 <code>.egg</code> 或 <code>.py</code> 檔案清單，以放置在 <code>PYTHONPATH</code> 適用於 Python 應用程式的中。	NULL

下表列出預設 Spark 提交參數。

預設 Spark 提交參數

金鑰	Description	預設值
<code>archives</code>	Spark 擷取到每個執行器工作目錄的以逗號分隔的封存清單。	NULL

金鑰	Description	預設值
class	應用程式的主要類別（適用於 Java 和 Scala 應用程式）。	NULL
conf	任意 Spark 組態屬性。	NULL
driver-cores	驅動程式使用的核心數量。	4
driver-memory	驅動程式使用的記憶體量。	14G
executor-cores	每個執行器使用的核心數量。	4
executor-memory	執行器使用的記憶體數量。	14G
files	以逗號分隔的檔案清單，放置在每個執行器的工作目錄中。您可以使用存取執行器中這些檔案的檔案路徑 <code>SparkFiles.get(fileName)</code> 。	NULL
jars	要包含在驅動程式和執行器 classpaths 上的逗號分隔 jar 清單。	NULL
num-executors	要啟動的執行器數目。	3
py-files	要放置在 PYTHONPATH 適用於 Python 應用程式的上的 .zip、.egg 或 .py 檔案的逗號分隔清單。	NULL
verbose	開啟其他偵錯輸出的選項。	NULL

資源組態最佳實務

透過 StartJobRun API 設定驅動程式和執行器資源

Note

如果指定 Spark 驅動程式和執行器核心和記憶體屬性，則必須在 StartJobRun API 請求中直接指定。

以此方式設定您的資源，可確保 EMR Serverless 可以在執行任務之前配置正確的資源。這與使用者指令碼中提供的設定相反，例如 .py 或 .jar 檔案中的評估太晚，因為驅動程式和執行器工作者有時會在指令碼執行開始之前預先佈建。在任務提交期間，有兩種支援的方法來設定這些資源：

選項 1：使用 sparkSubmitParameters

```
"jobDriver": {
  "sparkSubmit": {
    "entryPoint": "s3://your-script-path.py",
    "sparkSubmitParameters": "-conf spark.driver.memory=4g \
-conf spark.driver.cores=2 \
-conf spark.executor.memory=8g \
-conf spark.executor.cores=4"
  }
}
```

選項 2：使用 configurationOverrides 進行 spark-defaults 分類

```
"configurationOverrides": {
  "applicationConfiguration": [
    {
      "classification": "spark-defaults",
      "properties": {
        "spark.driver.memory": "4g",
        "spark.driver.cores": "2",
        "spark.executor.memory": "8g",
        "spark.executor.cores": "4"
      }
    }
  ]
}
```

Spark 範例

下列範例示範如何使用 StartJobRun API 來執行 Python 指令碼。如需使用此範例的end-to-end教學課程，請參閱 [Amazon EMR Serverless 入門](#)。您可以在 [EMR Serverless Samples](#) GitHub 儲存庫中找到有關如何執行 PySpark 任務和新增 Python 相依性的其他範例。

```
aws emr-serverless start-job-run \  
  --application-id application-id \  
  --execution-role-arn job-role-arn \  
  --job-driver '{  
    "sparkSubmit": {  
      "entryPoint": "s3://us-east-1.elasticmapreduce/emr-containers/samples/  
wordcount/scripts/wordcount.py",  
      "entryPointArguments": ["s3://amzn-s3-demo-destination-bucket/  
wordcount_output"],  
      "sparkSubmitParameters": "--conf spark.executor.cores=1 --conf  
spark.executor.memory=4g --conf spark.driver.cores=1 --conf spark.driver.memory=4g --  
conf spark.executor.instances=1"  
    }  
  }'
```

下列範例示範如何使用 StartJobRun API 來執行 Spark JAR。

```
aws emr-serverless start-job-run \  
  --application-id application-id \  
  --execution-role-arn job-role-arn \  
  --job-driver '{  
    "sparkSubmit": {  
      "entryPoint": "/usr/lib/spark/examples/jars/spark-examples.jar",  
      "entryPointArguments": ["1"],  
      "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi --conf  
spark.executor.cores=4 --conf spark.executor.memory=20g --conf spark.driver.cores=4 --  
conf spark.driver.memory=8g --conf spark.executor.instances=1"  
    }  
  }'
```

執行 EMR Serverless 任務時使用 Hive 組態

您可以在 `type` 參數設定為 `hive` 的應用程式上執行 Hive 任務HIVE。任務必須與 Amazon EMR 發行版本相容的 Hive 版本相容。例如，當您在具有 Amazon EMR 6.6.0 版的應用程式上執行任務時，您

的任務必須與 Apache Hive 3.1.2 相容。如需每個版本的應用程式版本資訊，請參閱 [Amazon EMR Serverless 發行版本](#)。

Hive 任務參數

當您使用 [StartJobRun API](#) 執行 Hive 任務時，請指定下列參數。

必要參數

- [Hive 任務執行時間角色](#)
- [Hive 任務驅動程式參數](#)
- [Hive 組態覆寫參數](#)

Hive 任務執行時間角色

使用 `executionRoleArn` 為應用程式用來執行 Hive 任務的 IAM 角色指定 ARN。此角色必須包含下列許可：

- 從資料所在的 S3 儲存貯體或其他資料來源讀取
- 從 Hive 查詢檔案和 init 查詢檔案所在的 S3 儲存貯體或字首讀取
- 讀取和寫入 Hive Scratch 目錄和 Hive Metastore 倉儲目錄所在的 S3 儲存貯體
- 寫入您要寫入最終輸出的 S3 儲存貯體
- 將日誌寫入 `S3MonitoringConfiguration` 指定的 S3 儲存貯體或字首
- 如果您使用 KMS 金鑰來加密 S3 儲存貯體中的資料，則存取 KMS 金鑰
- 存取 AWS Glue Data Catalog

如果您的 Hive 任務讀取資料或從其他資料來源寫入資料，請在此 IAM 角色中指定適當的許可。如果您未將這些許可提供給 IAM 角色，您的任務可能會失敗。如需詳細資訊，請參閱 [Amazon EMR Serverless 的任務執行期角色](#)。

Hive 任務驅動程式參數

使用 `jobDriver` 為任務提供輸入。任務驅動程式參數只接受您要執行之任務類型的一個值。當您將指定 `hive` 為任務類型時，EMR Serverless 會將 Hive 查詢傳遞給 `jobDriver` 參數。Hive 任務具有下列參數：

- **query** – 這是 Amazon S3 中您要執行之 Hive 查詢檔案的參考。

- **parameters** – 這些是您要覆寫的其他 Hive 組態屬性。若要覆寫屬性，請將它們以 `property=value` 的形式傳遞至此參數 `--hiveconf`。若要覆寫變數，請將它們以 `key=value` 的形式傳遞至此參數 `--hivevar`。
- **initQueryFile** – 這是 init Hive 查詢檔案。Hive 會在查詢之前執行此檔案，並可用來初始化資料表。

Hive 組態覆寫參數

使用 **configurationOverrides** 覆寫監控層級和應用程式層級組態屬性。此參數接受具有下列兩個欄位的 JSON 物件：

- **monitoringConfiguration** – 使用此欄位指定您希望 EMR Serverless 任務存放 Hive 任務日誌的 Amazon S3 URL (`s3MonitoringConfiguration`)。請確定您建立此儲存貯體的方式 AWS 帳戶與託管您應用程式的儲存貯體相同，且與您執行任務 AWS 區域的儲存貯體相同。
- **applicationConfiguration** – 您可以在此欄位中提供組態物件，以覆寫應用程式的預設組態。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。

Note

可用的組態分類會因特定 EMR Serverless 版本而有所不同。例如，自訂 `Log4j spark-driver-log4j2` 和 `spark-executor-log4j2` 僅適用於 6.8.0 版和更新版本。

如果您在應用程式覆寫和 Hive 參數中傳遞相同的組態，Hive 參數會優先。下列清單會將組態從最高優先順序排名為最低優先順序。

- 您使用 `--hiveconf` 作為 Hive 參數一部分提供的組態 `property=value`。
- 當您啟動任務時，您作為應用程式一部分提供的組態會覆寫。
- 您在建立應用程式 `runtimeConfiguration` 時作為的一部分提供的組態。
- Amazon EMR 為發行版本指派的最佳化組態。
- 應用程式的預設開放原始碼組態。

如需在應用程式層級宣告組態，以及在任務執行期間覆寫組態的詳細資訊，請參閱 [EMR Serverless 的預設應用程式組態](#)。

Hive 任務屬性

下表列出當您提交 Hive 任務時所設定的強制性屬性。

強制性 Hive 任務屬性

設定	Description
<code>hive.exec.scratchdir</code>	EMR Serverless 在 Hive 任務執行期間建立暫存檔案的 Amazon S3 位置。
<code>hive.metastore.warehouse.dir</code>	Hive 中受管資料表資料庫的 Amazon S3 位置。

下表列出您在提交 Hive 任務時可覆寫的選用 Hive 屬性及其預設值。

選用 Hive 屬性和預設值

設定	Description	預設值
<code>fs.s3.customAWSCredentialsProvider</code>	您要使用的 AWS 登入資料提供者。	<code>com.amazonaws.auth.DefaultAWSCredentialsProviderChain</code>
<code>fs.s3a.aws.credentials.provider</code>	您要搭配 S3A 檔案系統使用的 AWS 登入資料提供者。	<code>com.amazonaws.auth.DefaultAWSCredentialsProviderChain</code>
<code>hive.auto.convert.join</code>	根據輸入檔案大小，開啟常見聯結自動轉換至 mapjoins 的選項。	TRUE
<code>hive.auto.convert.join.noconditionaltask</code>	當 Hive 根據輸入檔案大小將常見聯結轉換為 mapjoin 時開啟最佳化的選項。	TRUE
<code>hive.auto.convert.join.noconditionaltask.size</code>	聯結會直接轉換為低於此大小的映射聯結。	最佳值是根據 Tez 任務記憶體計算

設定	Description	預設值
hive.cbo.enable	使用 Calcite 架構開啟成本型最佳化的選項。	TRUE
hive.cli.tez.session.async	在 Hive 查詢編譯時啟動背景 Tez 工作階段的選項。當設定為 <code>false</code> ，Tez AM 會在 Hive 查詢編譯之後啟動。	TRUE
hive.compute.query.using.stats	啟用 Hive 以使用中繼存放區中存放的統計資料回答特定查詢的選項。對於基本統計資料，請將 <code>hive.stats.autogather</code> 設定為 TRUE。如需更進階的查詢集合，請執行 <code>analyze table queries</code> 。	TRUE
hive.default.fileformat	CREATE TABLE 陳述式的預設檔案格式。如果您在 CREATE TABLE 命令 STORED AS [FORMAT] 中指定，您可以明確覆寫此項目。	TEXTFILE
hive.driver.cores	用於 Hive 驅動程式程序的核心數量。	2
hive.driver.disk	Hive 驅動程式的磁碟大小。	20G
hive.driver.disk.type	Hive 驅動程式的磁碟類型。	標準
hive.tez.disk.type	Tez 工作者的磁碟大小。	標準
hive.driver.memory	每個 Hive 驅動程式程序要使用的記憶體量。Hive CLI 和 Tez Application Master 與 20% 的前端空間平均共用此記憶體。	6G

設定	Description	預設值
hive.emr-serverless.launch.env.[<i>KEY</i>]	在所有 Hive 特定程序中設定 <i>KEY</i> 環境變數的選項，例如 Hive 驅動程式、Tez AM 和 Tez 任務。	
hive.exec.dynamic.partition	在 DML/DDL 中開啟動態分割區的選項。	TRUE
hive.exec.dynamic.partition.mode	指定您要使用嚴格模式或非嚴格模式的選項。在嚴格模式下，請指定至少一個靜態分割區，以防您不小心覆寫所有分割區。在非嚴格模式中，所有分割區都可以是動態的。	strict
hive.exec.max.dynamic.partitions	Hive 建立的動態分割區總數上限。	1000
hive.exec.max.dynamic.partitions.per.node	Hive 在每個映射器和縮減器節點中建立的動態分割區數目上限。	100
hive.exec.orc.split.strategy	預期下列其中一個值：BI、ETL 或 HYBRID。這不是使用者層級組態。BI 指定您想要花較少的時間在分割產生中，而不是查詢執行。ETL 指定您想要花更多時間在分割產生中。指定根據啟發式HYBRID選擇上述策略。	HYBRID
hive.exec.reducers.bytes.per.reducer	每個縮減器的大小。預設值為 256 MB。如果輸入大小為 1G，任務會使用 4 個縮減器。	256000000

設定	Description	預設值
hive.exec.reducers.max	減少程式的數量上限。	256
hive.exec.stagingdir	存放 Hive 在資料表位置和 hive.exec.scratchdir 屬性中指定之暫存目錄位置內建立之暫存檔案的目錄名稱。	.hive-staging
hive.fetch.task.conversion	預期下列其中一個值：NONE、MINIMAL 或 MORE。Hive 可以將選取查詢轉換為單一 FETCH 任務。這可將延遲降至最低。	MORE
hive.groupby.position.alias	讓 Hive 在 GROUP BY 陳述式中使用資料欄位置別名的選項。	FALSE
hive.input.format	預設輸入格式。HiveInputFormat 如果您遇到問題，請將設定為 CombineHiveInputFormat。	org.apache.hadoop.hive ql.io.CombineHiveInputFormat
hive.log.explain.output	開啟 Hive 日誌中任何查詢延伸輸出說明的選項。	FALSE
hive.log.level	Hive 記錄層級。	INFO
hive.mapred.reduce.tasks.speculative.execution	開啟縮減器推測啟動的選項。僅支援 Amazon EMR 6.10.x 及更低版本。	TRUE
hive.max-task-containers	並行容器的數量上限。設定的映射器記憶體會乘以此值，以判斷可分割運算和任務先佔用量的可用記憶體。	1000

設定	Description	預設值
hive.merge.mapfiles	在僅映射任務結束時，導致小型檔案合併的選項。	TRUE
hive.merge.size.pertask	任務結束時合併檔案的大小。	256000000
hive.merge.tezfiles	在 Tez DAG 結束時開啟小型檔案合併的選項。	FALSE
hive.metastore.client.factory.class	產生實作IMetaStoreClient 界面之物件的原廠類別名稱。	com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory
hive.metastore.glue.catalogid	如果 AWS Glue Data Catalog 充當中繼存放區，但在與任務不同的 AWS 帳戶中執行，則為任務執行 AWS 帳戶所在的 ID。	NULL
hive.metastore.uris	中繼存放區用戶端用來連線至遠端中繼存放區的偏離 URI。	NULL
hive.optimize.ppd	開啟述詞下推的選項。	TRUE
hive.optimize.ppd.storage	開啟述詞下推至儲存處理常式的選項。	TRUE
hive.orderby.position.alias	讓 Hive 在ORDER BY陳述式中使用資料欄位置別名的選項。	TRUE
hive.prewarm.enabled	開啟 Tez 容器預熱的選項。	FALSE
hive.prewarm.numcontainers	要為 Tez 預熱的容器數量。	10

設定	Description	預設值
<code>hive.stats.autogather</code>	讓 Hive 在 INSERT OVERWRITE 命令期間自動收集基本統計資料的選項。	TRUE
<code>hive.stats.fetch.column.stats</code>	關閉從中繼存放區擷取資料欄統計資料的選項。當資料欄數量很高時，擷取資料欄統計資料可能會很昂貴。	FALSE
<code>hive.stats.gather.num.threads</code>	<code>partialscan</code> 和 <code>noscan</code> 分析命令用於分割資料表的執行緒數量。這僅適用於實作的檔案格式 <code>StatsProvidingRecordReader</code> (例如 ORC)。	10
<code>hive.strict.checks.cartesian.product</code>	開啟嚴格笛卡爾聯結檢查的選項。這些檢查不允許笛卡兒產品 (交叉聯結)。	FALSE
<code>hive.strict.checks.type.safety</code>	開啟嚴格類型安全檢查並關閉 <code>bigint</code> 與 <code>string</code> 和比較的選項 <code>double</code> 。	TRUE
<code>hive.support.quoted.identifiers</code>	預期值為 NONE 或 COLUMN。NONE 表示只有英數字元和底線字元在識別符中有效。COLUMN 表示資料欄名稱可包含任何字元。	COLUMN
<code>hive.tez.auto.reducer.parallelism</code>	開啟 Tez 自動減少程式平行處理功能的選項。Hive 仍會估計資料大小並設定平行處理估計值。Tez 會取樣來源頂點的輸出大小，並視需要在執行時間調整預估值。	TRUE

設定	Description	預設值
hive.tez.container.size	每個 Tez 任務程序要使用的記憶體量。	6144
hive.tez.cpu.vcores	要用於每個 Tez 任務的核心數量。	2
hive.tez.disk.size	每個任務容器的磁碟大小。	20G
hive.tez.input.format	Tez AM 中分割產生的輸入格式。	org.apache.hadoop.hive ql.io.HiveInputFormat
hive.tez.min.partition.factor	當您開啟自動減少程式平行處理時，Tez 指定的減少程式下限。	0.25
hive.vectorized.execution.enabled	開啟查詢執行的向量化模式的選項。	TRUE
hive.vectorized.execution.reduce.enabled	開啟查詢執行縮減端之向量化模式的選項。	TRUE
javax.jdo.option.ConnectionDriverName	JDBC 中繼存放區的驅動程式類別名稱。	org.apache.derby.jdbc.EmbeddedDriver
javax.jdo.option.ConnectionPassword	與中繼存放區資料庫相關聯的密碼。	NULL
javax.jdo.option.ConnectionURL	JDBC 中繼存放區的 JDBC 連線字串。	jdbc:derby:;databaseName=metastore_db;create=true
javax.jdo.option.ConnectionUserName	與中繼存放區資料庫相關聯的使用者名稱。	NULL

設定	Description	預設值
<code>mapreduce.input.fileinputformat.split.maxsize</code>	當您的輸入格式為 <code>org.apache.hadoop.hive ql.io.CombineHiveInputFormat</code> 時，分割運算期間分割的大小上限。值 0 表示沒有限制。	0
<code>tez.am.dag.cleanup.on.completion</code>	當 DAG 完成時，開啟隨機資料清除的選項。	TRUE
<code>tez.am.emr-serverless.launch.env.[KEY]</code>	在 Tez AM 程序中設定 <code>KEY</code> 環境變數的選項。對於 Tez AM，此值會覆寫該 <code>hive.emr-serverless.launch.env.[KEY]</code> 值。	
<code>tez.am.log.level</code>	EMR Serverless 傳遞至 Tez 應用程式主要的根記錄層級。	INFO
<code>tez.am.sleep.time.before.exit.millis</code>	EMR Serverless 應在 AM 關閉請求後的這段時間之後推送 ATS 事件。	0
<code>tez.am.speculation.enabled</code>	導致推測啟動較慢任務的選項。這有助於在某些任務因為機器故障或速度變慢而執行速度變慢時減少任務延遲。僅支援 Amazon EMR 6.10.x 及更低版本。	FALSE
<code>tez.am.task.max.failed.attempts</code>	在任務失敗之前，特定任務可能失敗的嘗試次數上限。此數字不會計入手動終止的嘗試次數。	3

設定	Description	預設值
<code>tez.am.vertex.cleanup.height</code>	如果所有相依頂點都完成，Tez AM 會刪除頂點隨機播放資料的距離。當值為 0 時，此功能會關閉。Amazon EMR 6.8.0 版及更新版本支援此功能。	0
<code>tez.client.asynchronous-stop</code>	讓 EMR Serverless 在結束 Hive 驅動程式之前推送 ATS 事件的選項。	FALSE
<code>tez.grouping.max-size</code>	分組分割的大小上限（以位元組為單位）。此限制可防止過大的分割。	1073741824
<code>tez.grouping.min-size</code>	分組分割的大小下限（以位元組為單位）。此限制可防止太多小分割。	16777216
<code>tez.runtime.io.sort.mb</code>	當 Tez 排序輸出時，軟緩衝區的大小會排序。	最佳值是根據 Tez 任務記憶體計算
<code>tez.runtime.unordered.output.buffer.size-mb</code>	如果 Tez 未直接寫入磁碟，要使用的緩衝區大小。	最佳值是根據 Tez 任務記憶體計算

設定	Description	預設值
<code>tez.shuffle-vertex-manager.max-src-fraction</code>	在 EMR Serverless 排程目前頂點的所有任務之前（在 ScatterGather 連線的情況下），必須完成的來源任務部分。在目前頂點上準備好排程的任務數量，會在 <code>min-fraction</code> 和之間線性擴展 <code>max-fraction</code> 。這會預設為預設值或 <code>tez.shuffle-vertex-manager.min-src-fraction</code> ，以較大者為準。	0.75
<code>tez.shuffle-vertex-manager.min-src-fraction</code>	在 EMR Serverless 排程目前頂點的任務之前（在 ScatterGather 連線的情況下）必須完成的來源任務部分。	0.25
<code>tez.task.emr-serverless.launch.env.[KEY]</code>	在 Tez 任務程序中設定 KEY 環境變數的選項。對於 Tez 任務，此值會覆寫該 <code>hive.emr-serverless.launch.env.[KEY]</code> 值。	
<code>tez.task.log.level</code>	EMR Serverless 傳遞給 Tez 任務的根記錄層級。	INFO
<code>tez.yarn.ats.event.flush.timeout.millis</code>	AM 在關閉之前應等待事件排清的時間上限。	300000

Hive 任務範例

下列程式碼範例示範如何使用 `StartJobRun` API 執行 Hive 查詢。

```
aws emr-serverless start-job-run \  
  --application-id application-id \  
  --execution-role-arn job-role-arn \  
  --job-driver '{  
    "hive": {  
      "query": "s3://amzn-s3-demo-bucket/emr-serverless-hive/query/hive-  
query.sql",  
      "parameters": "--hiveconf hive.log.explain.output=false"  
    }  
  }' \  
  --configuration-overrides '{  
    "applicationConfiguration": [{  
      "classification": "hive-site",  
      "properties": {  
        "hive.exec.scratchdir": "s3://amzn-s3-demo-bucket/emr-serverless-hive/  
hive/scratch",  
        "hive.metastore.warehouse.dir": "s3://amzn-s3-demo-bucket/emr-  
serverless-hive/hive/warehouse",  
        "hive.driver.cores": "2",  
        "hive.driver.memory": "4g",  
        "hive.tez.container.size": "4096",  
        "hive.tez.cpu.vcores": "1"  
      }  
    }  
  ]}  
'
```

您可以在 [EMR Serverless Samples](#) GitHub 儲存庫中找到如何執行 Hive 任務的其他範例。

EMR Serverless 任務彈性

EMR Serverless 7.1.0 版和更新版本包含對任務彈性的支援，因此它會自動重試任何失敗的任務，而無需您進行任何手動輸入。任務彈性的另一個好處是，如果 AZ 遇到任何問題，EMR Serverless 會將任務執行移至不同的可用區域 (AZ)。

若要啟用任務的任務彈性，請為您的任務設定重試政策。重試政策可確保 EMR Serverless 在任何時間點失敗時自動重新啟動任務。批次和串流任務支援重試政策，因此請根據您的使用案例自訂任務彈性。下表比較批次和串流任務的任務彈性行為和差異。

	批次任務	串流任務
預設行為	不會重新執行任務。	當應用程式在執行任務時建立檢查點時，一律重試執行任務。
重試點	批次任務沒有檢查點，因此 EMR Serverless 一律會從頭重新執行任務。	串流任務支援檢查點，因此請設定串流查詢，以儲存執行時間狀態並進入 Amazon S3 中的檢查點位置。EMR Serverless 會從檢查點繼續任務執行。如需詳細資訊，請參閱 Apache Spark 文件中的 使用檢查點從失敗復原 。
重試嘗試次數上限	允許最多 10 次重試。	串流任務具有內建的防撞控制，因此如果任務在一小時後繼續失敗，應用程式會停止重試任務。一小時內的預設重試次數為 5 次嘗試。您可以將此重試次數設定為介於 1 或 10 之間。您無法自訂最大嘗試次數。值 1 表示沒有重試。

當 EMR Serverless 嘗試重新執行任務時，也會以嘗試次數為任務編製索引，因此會在任務嘗試期間追蹤任務的生命週期。

使用 EMR Serverless API 操作或 AWS CLI 來變更任務彈性或存取與任務彈性相關的資訊。如需詳細資訊，請參閱 [EMR Serverless API 指南](#)。

根據預設，EMR Serverless 不會重新執行批次任務。若要啟用批次任務的重試，請在開始批次任務執行時設定 `maxAttempts` 參數。`maxAttempts` 參數僅適用於批次任務。預設值為 1，這表示不重新執行任務。接受的值為 1 到 10，包含。

下列範例示範如何在開始任務執行時指定最多 10 次嘗試。

```
aws emr-serverless start-job-run
```

```

--application-id <APPLICATION_ID> \
--execution-role-arn <JOB_EXECUTION_ROLE> \
--mode 'BATCH' \
--retry-policy '{
  "maxAttempts": 10
}' \
--job-driver '{
  "sparkSubmit": {
    "entryPoint": "/usr/lib/spark/examples/jars/spark-examples-does-not-
exist.jar",
    "entryPointArguments": ["1"],
    "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi"
  }
}'

```

如果串流任務失敗，EMR Serverless 會無限期重試。若要防止因重複無法復原的失敗而發生當機，請使用 `maxFailedAttemptsPerHour` 設定串流任務重試的當機預防控制。此參數可讓您指定 EMR Serverless 停止重試前一小時允許的失敗嘗試次數上限。預設值為 5。接受的值為 1 到 10，包含。

```

aws emr-serverless start-job-run
--application-id <APPLICATION_ID> \
--execution-role-arn <JOB_EXECUTION_ROLE> \
--mode 'STREAMING' \
--retry-policy '{
  "maxFailedAttemptsPerHour": 7
}' \
--job-driver '{
  "sparkSubmit": {
    "entryPoint": "/usr/lib/spark/examples/jars/spark-examples-does-not-
exist.jar",
    "entryPointArguments": ["1"],
    "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi"
  }
}'

```

您也可以使用其他任務執行 API 操作來取得任務的相關資訊。例如，搭配 `GetJobRun` 操作使用 `attempt` 參數，以取得特定任務嘗試的詳細資訊。如果您未包含 `attempt` 參數，操作會傳回最新嘗試的相關資訊。

```

aws emr-serverless get-job-run \
--job-run-id <job-run-id> \
--application-id <application-id> \

```

```
--attempt 1
```

ListJobRunAttempts 操作會傳回與任務執行相關的所有嘗試的相關資訊。

```
aws emr-serverless list-job-run-attempts \  
  --application-id application-id \  
  --job-run-id job-run-id
```

GetDashboardForJobRun 操作會建立並傳回 URL，以使用存取任務執行的應用程式 UIs。attempt 參數可讓您取得特定嘗試的 URL。如果您未包含 attempt 參數，操作會傳回最新嘗試的相關資訊。

```
aws emr-serverless get-dashboard-for-job-run \  
  --application-id application-id \  
  --job-run-id job-run-id \  
  --attempt 1
```

使用重試政策監控作業

任務彈性支援也會新增事件 EMR Serverless 任務執行重試。EMR Serverless 會在任務的每次重試時發佈此事件。您可以使用此通知來追蹤任務的重試。如需事件的詳細資訊，請參閱 [Amazon EventBridge 事件](#)。

使用重試政策記錄

每次 EMR Serverless 重試任務時，嘗試都會產生自己的日誌集。為了確保 EMR Serverless 可以將這些日誌成功交付到 Amazon S3 和 Amazon CloudWatch，而不會覆寫任何日誌，EMR Serverless 會將字首新增至 S3 日誌路徑格式和 CloudWatch 日誌串流名稱，以包含任務的嘗試次數。

以下是 格式的範例。

```
'/applications/<applicationId>/jobs/<jobId>/attempts/<attemptNumber>/'.
```

此格式可確保 EMR Serverless 將每次嘗試任務的所有日誌發佈到 Amazon S3 和 CloudWatch 中自己的指定位置。如需詳細資訊，請參閱 [儲存日誌](#)。

Note

EMR Serverless 只會將此字首格式用於所有串流任務和任何已啟用重試的批次任務。

EMR Serverless 的中繼存放區組態

Hive 中繼存放區是集中位置，可存放資料表的結構資訊，包括結構描述、分割區名稱和資料類型。使用 EMR Serverless，會將此資料表中繼資料保留在可存取您任務的中繼存放區中。

Hive 中繼存放區有兩個選項：

- AWS Glue Data Catalog
- 外部 Apache Hive 中繼存放區

使用 AWS Glue Data Catalog 做為中繼存放區

您可以將 Spark 和 Hive 任務設定為使用 AWS Glue Data Catalog 作為其中繼存放區。當您需要持久性中繼存放區或不同應用程式、服務或共用的中繼存放區時，我們建議您使用此組態 AWS 帳戶。如需 Data Catalog 的詳細資訊，請參閱[填入 AWS Glue Data Catalog](#)。如需 AWS Glue 定價的相關資訊，請參閱[AWS Glue 定價](#)。

您可以將 EMR Serverless 任務設定為在 AWS 帳戶與應用程式相同的 或不同的 中使用 AWS Glue Data Catalog AWS 帳戶。

設定 AWS Glue Data Catalog

若要設定 Data Catalog，請選擇您要使用的 EMR Serverless 應用程式類型。

Spark

當您使用 EMR Studio 搭配 EMR Serverless Spark 應用程式執行任務時，AWS Glue Data Catalog 是預設中繼存放區。

當您使用 SDKs 或 AWS CLI，請在任務執行的 `sparkSubmit` 參數 `com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory` 中將 `spark.hadoop.hive.metastore.client.factory.class` 組態設定為 `com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory`。下列範例示範如何使用設定 Data Catalog AWS CLI。

```
aws emr-serverless start-job-run \
  --application-id application-id \
  --execution-role-arn job-role-arn \
  --job-driver '{
    "sparkSubmit": {
      "entryPoint": "s3://amzn-s3-demo-bucket/code/pyspark/
extreme_weather.py",
```

```

    "sparkSubmitParameters": "--conf
spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory
--conf spark.driver.cores=1 --conf spark.driver.memory=3g --conf
spark.executor.cores=4 --conf spark.executor.memory=3g"
  }
}'

```

或者，您可以在 Spark 程式碼 `SparkSession` 中建立新的時設定此組態。

```

from pyspark.sql import SparkSession

spark = (
    SparkSession.builder.appName("SparkSQL")
        .config(
            "spark.hadoop.hive.metastore.client.factory.class",
            "com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory",
        )
        .enableHiveSupport()
        .getOrCreate()
)

# we can query tables with SparkSQL
spark.sql("SHOW TABLES").show()

# we can also them with native Spark
print(spark.catalog.listTables())

```

Hive

對於 EMR Serverless Hive 應用程式，Data Catalog 是預設中繼存放區。也就是說，當您在 EMR Serverless Hive 應用程式上執行任務時，Hive 會在 AWS 帳戶與您應用程式相同的資料目錄中記錄中繼存放區資訊。您不需要虛擬私有雲端 (VPC) 即可使用 Data Catalog 做為中繼存放區。

若要存取 Hive 中繼存放區資料表，請新增設定 AWS Glue [的 IAM 許可中概述的必要 Glue AWS 政策](#)。

設定 EMR Serverless 和 Glue Data Catalog AWS 的跨帳戶存取

若要設定 EMR Serverless 的跨帳戶存取權，請先登入以下內容 AWS 帳戶：

- AccountA – 您已在 AWS 帳戶 其中建立 EMR Serverless 應用程式。
- AccountB – AWS 帳戶 包含您希望 EMR Serverless 任務執行存取的 AWS Glue Data Catalog。

1. 確定 中的管理員或其他授權身分將資源政策AccountB連接到 中的 Data CatalogAccountB。此政策授予AccountA特定跨帳戶許可，以對AccountB目錄中的資源執行操作。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabase",
        "glue:CreateDatabase",
        "glue:GetDataBases",
        "glue:CreateTable",
        "glue:GetTable",
        "glue:UpdateTable",
        "glue>DeleteTable",
        "glue:GetTables",
        "glue:GetPartition",
        "glue:GetPartitions",
        "glue:CreatePartition",
        "glue:BatchCreatePartition",
        "glue:GetUserDefinedFunctions"
      ],
      "Resource": [
        "arn:aws:glue:*:123456789012:catalog"
      ],
      "Sid": "AllowGLUEGetdatabase"
    }
  ]
}
```

2. 將 IAM 政策新增至 中的 EMR Serverless 任務執行期角色，AccountA讓該角色可以存取 中的 Data Catalog 資源AccountB。

JSON

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

```

    {
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabase",
        "glue:CreateDatabase",
        "glue:GetDataBases",
        "glue:CreateTable",
        "glue:GetTable",
        "glue:UpdateTable",
        "glue>DeleteTable",
        "glue:GetTables",
        "glue:GetPartition",
        "glue:GetPartitions",
        "glue:CreatePartition",
        "glue:BatchCreatePartition",
        "glue:GetUserDefinedFunctions"
      ],
      "Resource": [
        "arn:aws:glue:*:123456789012:catalog"
      ],
      "Sid": "AllowGLUEGetdatabase"
    }
  ]
}

```

3. 啟動您的任務執行。此步驟會因 AccountA 的 EMR Serverless 應用程式類型而略有不同。

Spark

在中傳遞 `spark.hadoop.hive.metastore.glue.catalogid` 屬性 `sparkSubmitParameters`，如下列範例所示。`AccountB-catalog-id` 將取代為中資料目錄的 ID `AccountB`。

```

aws emr-serverless start-job-run \
--application-id "application-id" \
--execution-role-arn "job-role-arn" \
--job-driver '{
  "sparkSubmit": {
    "entryPoint": "s3://amzn-s3-demo-bucket/scripts/test.py",
    "sparkSubmitParameters": "--conf
spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.A
--conf spark.hadoop.hive.metastore.glue.catalogid=AccountB-catalog-
id --conf spark.executor.cores=1 --conf spark.executor.memory=1g

```

```

--conf spark.driver.cores=1 --conf spark.driver.memory=1g --conf
spark.executor.instances=1"
    }
  }' \
--configuration-overrides '{
  "monitoringConfiguration": {
    "s3MonitoringConfiguration": {
      "logUri": "s3://amzn-s3-demo-bucket/logs/"
    }
  }
}'

```

Hive

在 `hive-site` 分類中設定 `hive.metastore.glue.catalogid` 屬性，如下列範例所示。將 *AccountB-catalog-id* 取代為 中資料目錄的 IDAccountB。

```

aws emr-serverless start-job-run \
--application-id "application-id" \
--execution-role-arn "job-role-arn" \
--job-driver '{
  "hive": {
    "query": "s3://amzn-s3-demo-bucket/hive/scripts/create_table.sql",
    "parameters": "--hiveconf hive.exec.scratchdir=s3://amzn-s3-demo-bucket/
hive/scratch --hiveconf hive.metastore.warehouse.dir=s3://amzn-s3-demo-bucket/
hive/warehouse"
  }
}' \
--configuration-overrides '{
  "applicationConfiguration": [{
    "classification": "hive-site",
    "properties": {
      "hive.metastore.glue.catalogid": "AccountB-catalog-id"
    }
  }]
}'

```

使用 Glue Data Catalog AWS 時的考量事項

您可以在 Hive 指令碼 ADD JAR 中使用 新增輔助 JARs。如需其他考量，請參閱 [使用 Glue Data Catalog AWS 時的考量](#)。

使用外部 Hive 中繼存放區

您可以設定 EMR Serverless Spark 和 Hive 任務連線到外部 Hive 中繼存放區，例如 Amazon Aurora 或 Amazon RDS for MySQL。本節說明如何設定 Amazon RDS Hive 中繼存放區、設定 VPC，以及設定 EMR Serverless 任務以使用外部中繼存放區。

建立外部 Hive 中繼存放區

1. 遵循建立 VPC 中的指示，使用私有子網路建立 Amazon Virtual Private Cloud (Amazon VPC)。 <https://docs.aws.amazon.com/vpc/latest/userguide/working-with-vpcs.html#Create-VPC>
2. 使用新的 Amazon VPC 和私有子網路建立 EMR Serverless 應用程式。當您使用 VPC 設定 EMR Serverless 應用程式時，它會先為您指定的每個子網路佈建彈性網路介面。然後，它會將您指定的安全群組連接到該網路介面。這可讓您的應用程式存取控制。如需如何設定 VPC 的詳細資訊，請參閱 [設定 EMR Serverless 應用程式的 VPC 存取以連線至資料](#)。
3. 在 Amazon VPC 的私有子網路中建立 MySQL 或 Aurora PostgreSQL 資料庫。如需有關如何建立 Amazon RDS 資料庫的資訊，請參閱 [建立 Amazon RDS 資料庫執行個體](#)。
4. 依照修改 [Amazon RDS 資料庫執行個體](#) 中的步驟，修改 MySQL 或 Aurora 資料庫的安全群組，以允許來自 EMR Serverless 安全群組的 JDBC 連線。從其中一個 EMR Serverless 安全群組將傳入流量規則新增至 RDS 安全群組。

Type	通訊協定	連接埠範圍	來源
所有 TCP	TCP	3306	emr-serverless-security-group

設定 Spark 選項

使用 JDBC

若要設定 EMR Serverless Spark 應用程式以根據 Amazon RDS for MySQL 或 Amazon Aurora MySQL 執行個體連線至 Hive 中繼存放區，請使用 JDBC 連線。在任務執行的 `spark-submit` 參數 `--jars` 中傳遞 `mariadb-connector-java.jar` 與。

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
```

```

--job-driver '{
  "sparkSubmit": {
    "entryPoint": "s3://amzn-s3-demo-bucket/scripts/spark-jdbc.py",
    "sparkSubmitParameters": "--jars s3://amzn-s3-demo-bucket/mariadb-
connector-java.jar
    --conf
    spark.hadoop.javax.jdo.option.ConnectionDriverName=org.mariadb.jdbc.Driver
    --conf spark.hadoop.javax.jdo.option.ConnectionUserName=<connection-user-
name>
    --conf spark.hadoop.javax.jdo.option.ConnectionPassword=<connection-
password>
    --conf spark.hadoop.javax.jdo.option.ConnectionURL=<JDBC-Connection-
string>
    --conf spark.driver.cores=2
    --conf spark.executor.memory=10G
    --conf spark.driver.memory=6G
    --conf spark.executor.cores=4"
  }
}' \
--configuration-overrides '{
  "monitoringConfiguration": {
    "s3MonitoringConfiguration": {
      "logUri": "s3://amzn-s3-demo-bucket/spark/logs/"
    }
  }
}'
}'

```

下列程式碼範例是與 Amazon RDS 上的 Hive 中繼存放區互動的 Spark 進入點指令碼。

```

from os.path import expanduser, join, abspath
from pyspark.sql import SparkSession
from pyspark.sql import Row
# warehouse_location points to the default location for managed databases and tables
warehouse_location = abspath('spark-warehouse')
spark = SparkSession \
    .builder \
    .config("spark.sql.warehouse.dir", warehouse_location) \
    .enableHiveSupport() \
    .getOrCreate()
spark.sql("SHOW DATABASES").show()
spark.sql("CREATE EXTERNAL TABLE `sampledb`.`sparknyctaxi`(`dispatching_base_num`
string, `pickup_datetime` string, `dropoff_datetime` string, `pulocationid` bigint,

```

```
`dolocationid` bigint, `sr_flag` bigint) STORED AS PARQUET LOCATION 's3://<s3 prefix>/nyctaxi_parquet/')
spark.sql("SELECT count(*) FROM sampledb.sparknyctaxi").show()
spark.stop()
```

使用 thrift 服務

您可以設定 EMR Serverless Hive 應用程式，以根據 Amazon RDS for MySQL 或 Amazon Aurora MySQL 執行個體連線至 Hive 中繼存放區。若要這樣做，請在現有 Amazon EMR 叢集的主節點上執行 thrift 伺服器。如果您已經擁有 Amazon EMR 叢集，且具有要用來簡化 EMR Serverless 任務組態的 thrift 伺服器，則此選項是理想的選擇。

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
  --job-driver '{
    "sparkSubmit": {
      "entryPoint": "s3://amzn-s3-demo-bucket/thriftscript.py",
      "sparkSubmitParameters": "--jars s3://amzn-s3-demo-bucket/mariadb-connector-java.jar
      --conf spark.driver.cores=2
      --conf spark.executor.memory=10G
      --conf spark.driver.memory=6G
      --conf spark.executor.cores=4"
    }
  }' \
  --configuration-overrides '{
    "monitoringConfiguration": {
      "s3MonitoringConfiguration": {
        "logUri": "s3://amzn-s3-demo-bucket/spark/logs/"
      }
    }
  }'
```

下列程式碼範例是進入點指令碼 (thriftscript.py)，使用 thrift 通訊協定連線到 Hive 中繼存放區。請注意，hive.metastore.uris 屬性需要設定為從外部 Hive 中繼存放區讀取。

```
from os.path import expanduser, join, abspath
from pyspark.sql import SparkSession
from pyspark.sql import Row
# warehouse_location points to the default location for managed databases and tables
warehouse_location = abspath('spark-warehouse')
```

```

spark = SparkSession \
  .builder \
  .config("spark.sql.warehouse.dir", warehouse_location) \
  .config("hive.metastore.uris", "thrift://thrift-server-host:thrift-server-port") \
  .enableHiveSupport() \
  .getOrCreate()
spark.sql("SHOW DATABASES").show()
spark.sql("CREATE EXTERNAL TABLE sampledb.`sparknyctaxi`(`dispatching_base_num`
  string, `pickup_datetime` string, `dropoff_datetime` string, `pulocationid` bigint,
  `dolocationid` bigint, `sr_flag` bigint) STORED AS PARQUET LOCATION 's3://<s3 prefix>/
nyctaxi_parquet/'")
spark.sql("SELECT * FROM sampledb.sparknyctaxi").show()
spark.stop()

```

設定 Hive 選項

使用 JDBC

如果您想要在 Amazon RDS MySQL 或 Amazon Aurora 執行個體上指定外部 Hive 資料庫位置，您可以覆寫預設中繼存放區組態。

Note

在 Hive 中，您可以同時對中繼存放區資料表執行多次寫入。如果您在兩個任務之間共用中繼存放區資訊，請確定您不會同時寫入相同的中繼存放區資料表，除非您寫入相同中繼存放區資料表的不同分割區。

在 `hive-site` 分類中設定下列組態，以啟用外部 Hive 中繼存放區。

```

{
  "classification": "hive-site",
  "properties": {
    "hive.metastore.client.factory.class":
"org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreClientFactory",
    "javax.jdo.option.ConnectionDriverName": "org.mariadb.jdbc.Driver",
    "javax.jdo.option.ConnectionURL": "jdbc:mysql://db-host:db-port/db-name",
    "javax.jdo.option.ConnectionUserName": "username",
    "javax.jdo.option.ConnectionPassword": "password"
  }
}

```

使用 thrift 伺服器

您可以設定 EMR Serverless Hive 應用程式，以根據 Amazon RDS for MySQL 或 Amazon Aurora MySQL Instance 連線至 Hive 中繼存放區。若要這樣做，請在現有 Amazon EMR 叢集的主節點上執行 thrift 伺服器。如果您已經有執行 thrift 伺服器的 Amazon EMR 叢集，並且想要使用 EMR Serverless 任務組態，則此選項是理想的選擇。

在 hive-site 分類中設定下列組態，讓 EMR Serverless 可以存取遠端漂移中繼存放區。請注意，您必須將 hive.metastore.uris 屬性設定為從外部 Hive 中繼存放區讀取。

```
{
  "classification": "hive-site",
  "properties": {
    "hive.metastore.client.factory.class":
"org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreClientFactory",
    "hive.metastore.uris": "thrift://thrift-server-host:thrift-server-port"
  }
}
```

在 EMR Serverless 上使用 AWS Glue 多目錄階層

您可以設定 EMR Serverless 應用程式以使用 AWS Glue 多目錄階層。下列範例示範如何搭配 AWS Glue 多目錄階層使用 EMR-S Spark。

若要進一步了解多目錄階層，請參閱[在 Glue Data Catalog 中使用 Amazon EMR AWS 上的 Spark 的多目錄階層](#)。

搭配 Iceberg 和 Glue Data Catalog 使用 Redshift 受管儲存 AWS (RMS)

以下示範如何設定 Spark 以與 Iceberg AWS 的 Glue Data Catalog 整合：

```
aws emr-serverless start-job-run \
  --application-id application-id \
  --execution-role-arn job-role-arn \
  --job-driver '{
    "sparkSubmit": {
      "entryPoint": "s3://amzn-s3-demo-bucket/myscript.py",
      "sparkSubmitParameters": "--conf spark.sql.catalog.nfgac_rms =
org.apache.iceberg.spark.SparkCatalog
      --conf spark.sql.catalog.rms.type=glue
      --conf spark.sql.catalog.rms.glue.id=Glue RMS catalog ID
      --conf spark.sql.defaultCatalog=rms
```

```

        --conf
spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions"
    }
}'

```

整合後來自目錄中資料表的範例查詢：

```
SELECT * FROM my_rms_schema.my_table
```

搭配 Iceberg REST API 和 Glue Data Catalog AWS 使用 Redshift 受管儲存 (RMS)

以下示範如何設定 Spark 以使用 Iceberg REST 目錄：

```

aws emr-serverless start-job-run \
--application-id application-id \
--execution-role-arn job-role-arn \
--job-driver '{
"sparkSubmit": {
"entryPoint": "s3://amzn-s3-demo-bucket/myscript.py",
"sparkSubmitParameters": "
--conf spark.sql.catalog.rms=org.apache.iceberg.spark.SparkCatalog
--conf spark.sql.catalog.rms.type=rest
--conf spark.sql.catalog.rms.warehouse=Glue RMS catalog ID
--conf spark.sql.catalog.rms.uri=Glue endpoint URI/iceberg
--conf spark.sql.catalog.rms.rest.sigv4-enabled=true
--conf spark.sql.catalog.rms.rest.signing-name=glue
--conf
spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions"
}
}'

```

目錄中資料表的範例查詢：

```
SELECT * FROM my_rms_schema.my_table
```

使用外部中繼存放區時的考量事項

- 您可以設定與 MariaDB JDBC 相容的資料庫做為中繼存放區。這些資料庫的範例包括 RDS for MariaDB、MySQL 和 Amazon Aurora。
- 中繼存放區不會自動初始化。如果您的中繼存放區未使用 Hive 版本的結構描述初始化，請使用 [Hive 結構描述工具](#)。

- EMR Serverless 不支援 Kerberos 身分驗證。您無法搭配 Kerberos 身分驗證與 EMR Serverless Spark 或 Hive 任務使用 thrift 中繼存放區伺服器。
- 您必須設定 VPC 存取，才能使用多目錄階層。

從 EMR Serverless 存取另一個 AWS 帳戶中的 S3 資料

您可以從一個 AWS 帳戶執行 Amazon EMR Serverless 任務，並將其設定為存取屬於另一個 AWS 帳戶的 Amazon S3 儲存貯體中的資料。此頁面說明如何從 EMR Serverless 設定 S3 的跨帳戶存取。

在 EMR Serverless 上執行的任務可以使用 S3 儲存貯體政策或擔任的角色，從不同的 AWS 帳戶存取 Amazon S3 中的資料。

先決條件

若要設定 Amazon EMR Serverless 的跨帳戶存取權，請在登入兩個 AWS 帳戶時完成任務：

- **AccountA** – 這是 AWS 您已建立 Amazon EMR Serverless 應用程式的帳戶。在您設定跨帳戶存取之前，請在此帳戶中備妥下列項目：
 - 您要執行任務的 Amazon EMR Serverless 應用程式。
 - 任務執行角色，具有在應用程式中執行任務所需的許可。如需詳細資訊，請參閱 [Amazon EMR Serverless 的任務執行期角色](#)。
- **AccountB** – 這是 AWS 包含您希望 Amazon EMR Serverless 任務存取之 S3 儲存貯體的帳戶。

使用 S3 儲存貯體政策來存取跨帳戶 S3 資料

若要 account B 從 存取 中的 S3 儲存貯體 account A，請將下列政策連接至 中的 S3 儲存貯體 account B。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ExamplePermissions1",
      "Effect": "Allow",
      "Principal": {
```

```

    "AWS": "arn:aws:iam::123456789012:root"
  },
  "Action": [
    "s3:ListBucket"
  ],
  "Resource": [
    "arn:aws:s3:::my-bucket-name"
  ]
},
{
  "Sid": "ExamplePermissions2",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::123456789012:root"
  },
  "Action": [
    "s3:PutObject",
    "s3:GetObject",
    "s3:DeleteObject"
  ],
  "Resource": [
    "arn:aws:s3:::my-bucket-name/*"
  ]
}
]
}

```

如需使用 S3 儲存貯體政策進行 S3 跨帳戶存取的詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的[範例 2：授予跨帳戶儲存貯體許可的儲存貯體擁有者](#)。

使用擔任的角色來存取跨帳戶 S3 資料

設定 Amazon EMR Serverless 跨帳戶存取權的另一種方法是使用 AWS Security Token Service (AWS STS) 中的 AssumeRole 動作。AWS STS 是一種全域 Web 服務，可讓您為使用者請求暫時、有限權限的登入資料。您可以使用使用建立的臨時安全登入資料，對 EMR Serverless 和 Amazon S3 進行 API 呼叫 AssumeRole。

下列步驟說明如何使用擔任的角色從 EMR Serverless 存取跨帳戶 S3 資料：

1. 在中建立 Amazon S3 儲存貯體、*cross-account-bucket* 體 AccountB。如需詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的[建立儲存貯體](#)。如果您想要跨帳戶存取

DynamoDB，也請在 中建立 DynamoDB 資料表 AccountB。如需詳細資訊，請參閱《[Amazon DynamoDB 開發人員指南](#)》中的 [建立 DynamoDB 資料表](#)。DynamoDB

2. 在 Cross-Account-Role-B 中建立可存取 *cross-account-bucket* 體的 AccountB IAM 角色。
 - a. 登入 AWS 管理主控台 並開啟位於 <https://console.aws.amazon.com/iam/> 的 IAM 主控台。
 - b. 選擇角色，並建立一個新角色 Cross-Account-Role-B。如需如何建立 IAM 角色的詳細資訊，請參閱《[IAM 使用者指南](#)》中的 [建立 IAM 角色](#)。
 - c. 建立 IAM 政策，指定 Cross-Account-Role-B 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 IAM 政策](#)。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:*"
      ],
      "Resource": [
        "arn:aws:s3:::cross-account-bucket",
        "arn:aws:s3:::cross-account-bucket/*"
      ],
      "Sid": "AllowS3"
    }
  ]
}
```

如果您需要 DynamoDB 存取，請建立 IAM 政策，指定存取跨帳戶 DynamoDB 資料表的許可。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 [Amazon DynamoDB：允許存取特定資料表](#)。

以下是允許存取 DynamoDB 資料表 的政策 CrossAccountTable。

JSON

```
{
```

```

"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:*"
    ],
    "Resource": [
      "arn:aws:dynamodb:*:123456789012:table/CrossAccountTable"
    ],
    "Sid": "AllowDYNAMODB"
  }
]
}

```

3. 編輯 Cross-Account-Role-B 角色的信任關係。

- a. 若要設定角色的信任關係，請在 IAM 主控台中為您步驟 2 中 Cross-Account-Role-B 建立的角色選擇信任關係索引標籤。
- b. 選取編輯信任關係。
- c. 新增下列政策文件。這可讓 Job-Execution-Role-A 中的 AccountA 擔任該 Cross-Account-Role-B 角色。

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSTSAssumerole",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/Job-Execution-Role-A"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

4. 在 Job-Execution-Role-A AccountA 中 AWS STS 授予擔任的 AssumeRole 許可 Cross-Account-Role-B。

- a. 在 AWS 帳戶的 IAM 主控台中 AccountA，選取 Job-Execution-Role-A。

- b. 將以下政策陳述式新增至 Job-Execution-Role-A 以允許 Cross-Account-Role-B 角色的 AssumeRole 動作。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": [
        "arn:aws:iam::123456789012:role/Cross-Account-Role-B"
      ],
      "Sid": "AllowSTSAssumerole"
    }
  ]
}
```

擔任的角色範例

使用單一擔任角色存取帳戶中的所有 S3 資源，或使用 Amazon EMR 6.11 及更高版本，設定多個 IAM 角色，以便在您存取不同的跨帳戶 S3 儲存貯體時擔任。

主題

- [使用一個擔任的角色存取 S3 資源](#)
- [使用多個擔任的角色存取 S3 資源](#)

使用一個擔任的角色存取 S3 資源

Note

當您將任務設定為使用單一擔任角色時，整個任務的所有 S3 資源都會使用該角色，包括 `entryPoint` 指令碼。

如果您想要使用單一擔任的角色來存取帳戶 B 中的所有 S3 資源，請指定下列組態：

1. 將 EMRFS 組態指定 `fs.s3.customAWSCredentialsProvider` 給 `com.amazonaws.emr.AssumeRoleAWSCredentialsProvider`。
2. 針對 Spark，使用 `spark.emr-serverless.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` 和 `spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` 在驅動程式和執行器上指定環境變數。
3. 針對 Hive，使用 `hive.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN`、`tez.am.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` 和 `tez.task.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` 在 Hive 驅動程式、Tez 應用程式主要容器和 Tez 任務容器上指定環境變數。

下列範例示範如何使用 擔任的角色，以跨帳戶存取啟動 EMR Serverless 任務執行。

Spark

下列範例示範如何使用 擔任的角色啟動具有 S3 跨帳戶存取權的 EMR Serverless Spark 任務執行。

```
aws emr-serverless start-job-run \
  --application-id application-id \
  --execution-role-arn job-role-arn \
  --job-driver '{
    "sparkSubmit": {
      "entryPoint": "entrypoint_location",
      "entryPointArguments": [":argument_1:", ":argument_2:"],
      "sparkSubmitParameters": "--conf spark.executor.cores=4 --conf
spark.executor.memory=20g --conf spark.driver.cores=4 --conf spark.driver.memory=8g
--conf spark.executor.instances=1"
    }
  }' \
  --configuration-overrides '{
    "applicationConfiguration": [{
      "classification": "spark-defaults",
      "properties": {
        "spark.hadoop.fs.s3.customAWSCredentialsProvider":
"com.amazonaws.emr.AssumeRoleAWSCredentialsProvider",
        "spark.emr-serverless.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
"arn:aws:iam::AccountB:role/Cross-Account-Role-B",
```

```

        "spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
"arn:aws:iam::AccountB:role/Cross-Account-Role-B"
    }
  ]]
}'

```

Hive

下列範例示範如何使用 擔任的角色來啟動具有 S3 跨帳戶存取權的 EMR Serverless Hive 任務執行。

```

aws emr-serverless start-job-run \
  --application-id application-id \
  --execution-role-arn job-role-arn \
  --job-driver '{
    "hive": {
      "query": "query_location",
      "parameters": "hive_parameters"
    }
  }' \
  --configuration-overrides '{
    "applicationConfiguration": [{
      "classification": "hive-site",
      "properties": {
        "fs.s3.customAWSCredentialsProvider":
"com.amazonaws.emr.serverless.credentialsprovider.AssumeRoleAWSCredentialsProvider",
        "hive.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
"arn:aws:iam::AccountB:role/Cross-Account-Role-B",
        "tez.am.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
"arn:aws:iam::AccountB:role/Cross-Account-Role-B",
        "tez.task.emr-
serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
"arn:aws:iam::AccountB:role/Cross-Account-Role-B"
      }
    }
  }'

```

使用多個擔任的角色存取 S3 資源

使用 EMR Serverless 6.11.0 版和更新版本，設定多個 IAM 角色，以便在您存取不同的跨帳戶儲存貯體時擔任。如果您想要存取帳戶 B 中具有不同擔任角色的不同 S3 資源，請在開始任務執行時使用下列組態：

1. 將 EMRFS 組態指定 `fs.s3.customAWSCredentialsProvider` 給 `com.amazonaws.emr.serverless.credentialsprovider.BucketLevelAssumeRoleCredentialsProvider`
2. 指定 EMRFS 組態 `fs.s3.bucketLevelAssumeRoleMapping`，以定義從 S3 儲存貯體名稱到帳戶 B 中 IAM 角色的映射。值的格式應該為 `bucket1->role1;bucket2->role2`。

例如，使用 `arn:aws:iam::AccountB:role/Cross-Account-Role-B-1` 存取儲存貯體 `bucket1`，並使用 `arn:aws:iam::AccountB:role/Cross-Account-Role-B-2` 存取儲存貯體 `bucket2`。下列範例示範如何透過多個擔任的角色，透過跨帳戶存取啟動 EMR Serverless 任務執行。

Spark

下列範例示範如何使用多個擔任的角色來建立 EMR Serverless Spark 任務執行。

```
aws emr-serverless start-job-run \
  --application-id application-id \
  --execution-role-arn job-role-arn \
  --job-driver '{
    "sparkSubmit": {
      "entryPoint": "entrypoint_location",
      "entryPointArguments": [":argument_1:", ":argument_2:"],
      "sparkSubmitParameters": "--conf spark.executor.cores=4 --conf
spark.executor.memory=20g --conf spark.driver.cores=4 --conf spark.driver.memory=8g
--conf spark.executor.instances=1"
    }
  }' \
  --configuration-overrides '{
    "applicationConfiguration": [{
      "classification": "spark-defaults",
      "properties": {
        "spark.hadoop.fs.s3.customAWSCredentialsProvider":
"com.amazonaws.emr.serverless.credentialsprovider.BucketLevelAssumeRoleCredentialsProvider"
        "spark.hadoop.fs.s3.bucketLevelAssumeRoleMapping":
"bucket1->arn:aws:iam::AccountB:role/Cross-Account-Role-B-1;bucket2-
>arn:aws:iam::AccountB:role/Cross-Account-Role-B-2"
      }
    }]
  }'
```

Hive

下列範例示範如何使用多個擔任的角色來建立 EMR Serverless Hive 任務執行。

```
aws emr-serverless start-job-run \  
  --application-id application-id \  
  --execution-role-arn job-role-arn \  
  --job-driver '{  
    "hive": {  
      "query": "query_location",  
      "parameters": "hive_parameters"  
    }  
  }' \  
  --configuration-overrides '{  
    "applicationConfiguration": [{  
      "classification": "hive-site",  
      "properties": {  
        "fs.s3.customAWSCredentialsProvider":  
        "com.amazonaws.emr.serverless.credentialsprovider.AssumeRoleAWSCredentialsProvider",  
        "fs.s3.bucketLevelAssumeRoleMapping": "bucket1-  
>arn:aws:iam::AccountB:role/Cross-Account-Role-B-1;bucket2-  
>arn:aws:iam::AccountB:role/Cross-Account-Role-B-2"  
      }  
    }]  
  }'
```

故障診斷 EMR Serverless 中的錯誤

使用以下資訊來協助診斷和修正使用 Amazon EMR Serverless 時發生的常見問題。

主題

- [錯誤：任務失敗，因為帳戶已達到可同時使用的最大 vCPU 服務限制。](#)
- [錯誤：任務失敗，因為應用程式已超過maximumCapacity設定。](#)
- [錯誤：由於無法配置工作者而導致任務失敗，因為應用程式已超過maximumCapacity。](#)
- [錯誤：S3 存取遭拒。請檢查所需 S3 資源上任務執行時間角色的 S3 存取許可。](#)
- [錯誤：ModuleNotFoundError：沒有名為 <module> 的模組。有關如何搭配 EMR Serverless 使用 python 程式庫的使用者指南。](#)
- [錯誤：無法擔任執行角色 <role name>，因為它不存在或未設定所需的信任關係。](#)

錯誤：任務失敗，因為帳戶已達到可同時使用的最大 vCPU 服務限制。

此錯誤表示由於帳戶已超過最大容量，EMR Serverless 無法提交任務。增加帳戶的最大容量。檢查 [EMR Serverless 服務配額](#) 的服務限制。

錯誤：任務失敗，因為應用程式已超過maximumCapacity設定。

此錯誤表示 EMR Serverless 無法提交任務，因為應用程式已超過設定的最大容量。增加應用程式的最大容量。

錯誤：由於無法配置工作者而導致任務失敗，因為應用程式已超過maximumCapacity。

此錯誤表示任務無法完成。無法配置工作者，因為應用程式已超過maximumCapacity設定。

錯誤：S3 存取遭拒。請檢查所需 S3 資源上任務執行時間角色的 S3 存取許可。

此錯誤表示您的任務無法存取您的 S3 資源。確認任務執行時間角色具有存取任務需要使用之 S3 資源的許可。若要進一步了解執行期角色，請參閱 [Amazon EMR Serverless 的任務執行期角色](#)。

錯誤：ModuleNotFoundError：沒有名為 <module> 的模組。有關如何搭配 EMR Serverless 使用 python 程式庫的使用者指南。

此錯誤表示 Python 模組不適用於 Spark 任務。檢查相依的 Python 程式庫是否可供任務使用。如需如何封裝 Python 程式庫的詳細資訊，請參閱 [搭配 EMR Serverless 使用 Python 程式庫](#)。

錯誤：無法擔任執行角色 <role name>，因為它不存在或未設定所需的信任關係。

此錯誤表示您為任務指定的任務執行期角色不存在，或該角色沒有 EMR Serverless 許可的信任關係。若要驗證 IAM 角色是否存在，並驗證您已正確設定角色的信任政策，請參閱 [中的指示 Amazon EMR Serverless 的任務執行期角色](#)。

啟用任務層級成本分配

任務層級成本分配可在個別任務執行層級啟用 EMR Serverless 的精細計費屬性，而不是彙總應用程式層級的所有成本。啟用時，您可以依與任務執行相關聯的特定任務執行 IDs 和標籤，篩選和追蹤 AWS Cost Explorer 和成本和用量報告，以便更清楚地了解提交的任務執行費用。

預設行為

預設不會啟用任務層級成本分配。

如何啟用或停用此功能

您可以在建立應用程式期間設定任務層級成本分配，或更新現有應用程式的任務層級成本分配。

建立新應用程式時指定 `jobLevelCostAllocation` 參數：

```
# Enable job-level cost allocation:
aws emr-serverless create-application \
  --name "my-application" \
  --release-label "emr-7.12.0" \
  --type "SPARK" \
  --job-level-cost-allocation-configuration '{
    "enabled": true
  }'

# Disable job-level cost allocation:
aws emr-serverless create-application \
  --name "my-application" \
  --release-label "emr-7.12.0" \
  --type "SPARK" \
  --job-level-cost-allocation-configuration '{
    "enabled": false
  }'
```

更新現有應用程式的 `jobLevelCostAllocationConfiguration` 參數：

```
# Enable job-level cost allocation:
aws emr-serverless update-application \
  --application-id <application-id> \
  --job-level-cost-allocation-configuration '{
    "enabled": true
  }'
```

```
}'  
  
# Disable job-level cost allocation:  
aws emr-serverless update-application \  
  --application-id <application-id> \  
  --job-level-cost-allocation-configuration '{  
    "enabled": false  
  }'  
}'
```

考量事項與限制

- 啟用任務層級成本分配不會追溯啟用功能之前完成的任務執行成本。啟用此功能後開始的任務執行會有精細的成本歸因。
- 只有在應用程式處於 CREATED 或 STOPPED 狀態時，才能更新任務層級成本分配參數。
- 啟用任務層級成本分配時，成本會歸因於個別任務執行，而非應用程式。若要在應用程式層級檢視彙總成本，您必須將一致的標籤（例如 application-name 或 application-id）套用至該應用程式內的所有任務執行，並在 Cost Explorer 或 Cost and Usage Reports 中依這些標籤進行篩選。

透過 Spark Connect 使用 Amazon EMR Serverless 執行互動式工作階段

使用 Amazon EMR 版本 `emr-7.13.0` 和更新版本，您可以從自我管理的 PySpark 用戶端連線至 Amazon EMR Serverless 應用程式，例如使用 EMR Serverless 工作階段 APIs 搭配 Apache Spark Connect 的 VS Code、PyCharm 和 Jupyter 筆記本。Spark Connect 使用用戶端伺服器架構，將您的應用程式程式碼與 Spark 驅動程式程序分離。您可以在本機 IDE 中開發和偵錯 PySpark 程式碼，同時在 EMR Serverless 運算上執行 Spark 操作。Spark Connect 提供下列優點：

- 從任何 PySpark 用戶端連線至 EMR Serverless，包括 VS Code、PyCharm 和 Jupyter 筆記本。
- 在 IDE 中設定中斷點並逐步完成 PySpark 程式碼，同時 DataFrames 遠端在生產規模資料上執行。

Spark Connect 工作階段是本機 PySpark 用戶端與在 Amazon EMR Serverless 上執行的 Spark 驅動程式之間的受管連線。當您啟動工作階段時，EMR Serverless 會代表您佈建 Spark 驅動程式和執行器。您的本機用戶端會將 DataFrame 和 SQL 操作傳送至驅動程式，而驅動程式會從遠端執行這些操作。工作階段會持續存在，直到您將其終止或達到閒置逾時為止，因此您可以以互動方式執行多個查詢，而無需重新啟動 Spark。每個工作階段都有自己的端點 URL 和身分驗證字符，您可用來連線。

所需的許可

除了存取 Amazon EMR Serverless 所需的許可之外，也請將下列許可新增至您的 IAM 角色，以存取 Spark Connect 端點和管理 Spark Connect 工作階段：

`emr-serverless:StartSession`

准許在您指定為的應用程式上建立 Spark Connect 工作階段 Resource。

`emr-serverless:GetSessionEndpoint`

准許擷取工作階段的 Spark Connect 端點 URL 和身分驗證字符。

`emr-serverless:GetSession`

准許取得工作階段的狀態。

`emr-serverless:ListSessions`

准許列出應用程式上的工作階段。

emr-serverless:TerminateSession

准許終止工作階段。

iam:PassRole

准許在建立 Spark Connect 工作階段時存取 IAM 執行角色。Amazon EMR Serverless 使用此角色來執行您的工作負載。

emr-serverless:GetResourceDashboard

准許產生 Spark UI URL，並提供工作階段日誌的存取權。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessApplicationLevelAccess",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:StartSession",
        "emr-serverless:ListSessions"
      ],
      "Resource": [
        "arn:aws:emr-serverless:region:account-id:/applications/application-id"
      ]
    },
    {
      "Sid": "EMRServerlessSessionLevelAccess",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:GetSession",
        "emr-serverless:GetSessionEndpoint",
        "emr-serverless:TerminateSession",
        "emr-serverless:GetResourceDashboard"
      ],
      "Resource": [
        "arn:aws:emr-serverless:region:account-id:/applications/application-id/
sessions/*"
      ]
    },
    {
      "Sid": "EMRServerlessRuntimeRoleAccess",
      "Effect": "Allow",
```

```
    "Action": [
      "iam:PassRole"
    ],
    "Resource": [
      "arn:aws:iam::account-id:role/EMRServerlessExecutionRole"
    ],
    "Condition": {
      "StringLike": {
        "iam:PassedToService": "emr-serverless.amazonaws.com"
      }
    }
  }
}
```

使用互動式工作階段

若要建立已啟用 Spark Connect 的應用程式並與其連線，請遵循下列步驟。

啟動 Spark Connect 工作階段

1. 使用 Spark Connect 工作階段建立應用程式。

```
aws emr-serverless create-application \  
  --type "SPARK" \  
  --name "spark-connect-app" \  
  --release-label emr-7.13.0 \  
  --interactive-configuration '{"sessionEnabled": true}'
```

2. Amazon EMR Serverless 建立應用程式後，如果您尚未啟用自動啟動以接受 Spark Connect 工作階段，請啟動應用程式。

```
aws emr-serverless start-application \  
  --application-id APPLICATION_ID
```

3. 使用下列命令來檢查應用程式的狀態。狀態變為 後STARTED，啟動工作階段。

```
aws emr-serverless get-application \  
  --application-id APPLICATION_ID
```

4. 使用授予資料存取權的 IAM 執行角色啟動工作階段。

```
aws emr-serverless start-session \
  --application-id APPLICATION_ID \
  --execution-role-arn arn:aws:iam::account-id:role/EMRServerlessExecutionRole
```

5. 使用 `get-session` API 監控工作階段狀態，並等待工作階段處於 `STARTED` 或 `IDLE` 狀態。

```
aws emr-serverless get-session \
  --application-id APPLICATION_ID \
  --session-id SESSION_ID
```

6. 擷取 Spark Connect 端點和身分驗證字符。傳回的端點 URL `GetSessionEndpoint` 不包含連接埠號碼。建構 `sc://連線` URL 時，您必須附加 `:443` - 例如 `sc://hostname:443/;use_ssl=true;x-aws-proxy-auth=token`。如果沒有它，PySpark 用戶端預設為連接埠 15002，無法在 EMR Serverless 上連線。

```
aws emr-serverless get-session-endpoint \
  --application-id APPLICATION_ID \
  --session-id SESSION_ID
```

回應包含端點 URL 和身分驗證字符：

```
{
  "endpoint": "ENDPOINT_URL",
  "authToken": "AUTH_TOKEN",
  "authTokenExpiresAt": "AUTH_TOKEN_EXPIRY_TIME"
}
```

7. 端點就緒後，從 PySpark 用戶端連線。在 EMR Serverless 應用程式上安裝符合 Spark 版本的 PySpark 用戶端，以及適用於 Python 的 AWS SDK。

```
# Match the PySpark version to your EMR Serverless release version (3.5.6 for
  emr-7.13.0)
pip install pyspark[connect]==3.5.6
pip install boto3
```

以下是啟動工作階段並直接將請求傳送至工作階段端點的範例 Python 指令碼：

```
import boto3
import time
```

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

client = boto3.client('emr-serverless', region_name='REGION')

APPLICATION_ID = 'APPLICATION_ID'
EXECUTION_ROLE = 'arn:aws:iam::account-id:role/EMRServerlessExecutionRole'

# Start the session
response = client.start_session(
    applicationId=APPLICATION_ID,
    executionRoleArn=EXECUTION_ROLE
)
session_id = response['sessionId']
print(f"Session {session_id} starting...")

# Wait for the session to be ready
while True:
    response = client.get_session(
        applicationId=APPLICATION_ID,
        sessionId=session_id
    )
    state = response['session']['state']
    print(f"Session state: {state}")
    if state in ('STARTED', 'IDLE'):
        break
    if state in ('FAILED', 'TERMINATED'):
        raise Exception(f"Session failed: {response['session'].get('stateDetails',
'Unknown error')}")
    time.sleep(5)

# Retrieve the Spark Connect endpoint and authentication token
response = client.get_session_endpoint(
    applicationId=APPLICATION_ID,
    sessionId=session_id
)

# Construct the authenticated remote URL
auth_token = response['authToken']
endpoint_url = response['endpoint']
connect_url = endpoint_url.replace("https://", "sc://", 1) + ":443/;use_ssl=true;"
connect_url += f"x-aws-proxy-auth={auth_token}"

# Start the Spark session
```

```
spark = SparkSession.builder.remote(connect_url).getOrCreate()
print(f"Connected. Spark version: {spark.version}")

# Run SQL
spark.sql("SELECT 1+1 AS result").show()

# Run DataFrame operations
df = spark.range(100).withColumn("squared", col("id") * col("id"))
df.show(10)
print(f"Count: {df.count()}")

# Stop the Spark session (disconnects the client only)
spark.stop()

# Terminate the EMR Serverless session to stop billing.
# spark.stop() only closes the local client connection. The remote session
# continues running and incurring charges until you explicitly terminate it
# or it reaches the idle timeout.
client.terminate_session(
    applicationId=APPLICATION_ID,
    sessionId=session_id
)
print(f"Session {session_id} terminated.")
```

若要存取工作階段的即時 Spark UI 或 Spark 歷史記錄伺服器，請使用 `GetResourceDashboard` API。

```
response = client.get_resource_dashboard(
    applicationId=APPLICATION_ID,
    resourceId=session_id,
    resourceType='SESSION'
)
response['url']
```

當工作階段處於作用中狀態時，URL 會開啟即時 Apache Spark UI，以即時監控查詢、階段和執行器。工作階段結束後，Spark 歷史記錄伺服器仍可透過 Amazon EMR Serverless 主控台進行工作階段後分析。

考量和限制

透過 Spark Connect 執行互動式工作負載時，請考慮下列事項。

- Amazon EMR Serverless 版本 `emr-7.13.0` 和更新版本支援 Spark Connect。
- 只有 Apache Spark 引擎支援 Spark Connect。
- Spark Connect 支援 PySpark 中的 DataFrame 和 SQL APIs。不支援 RDD 型 APIs。
- 身分驗證字符的時間限制為 1 小時。當字符過期時，gRPC 呼叫會失敗並出現身分驗證錯誤。呼叫 `GetSessionEndpoint` 以取得新的字符，並使用 `SparkSession` 更新的字符建立新的字符。
- 工作階段會在可設定的閒置逾時後結束。預設逾時設定為 1 小時。
- 根據預設，每個工作階段的硬性限制為 24 小時，之後即使它正在主動執行任務，也會自動終止。
- 根據預設，每個 EMR Serverless 應用程式最多支援 25 個並行工作階段。若要請求提高限制，請聯絡 AWS Support。
- 根據預設，`autoStopConfig` 會針對應用程式開啟。應用程式會在 15 分鐘後自動停止，沒有作用中的工作階段或任務執行。您可以在 `create-application` 或 `update-application` 請求中變更此組態。
- 為了獲得最佳的啟動體驗，請為驅動程式和執行器設定預先初始化的容量。
- 在啟動 EMR Serverless 工作階段之前，您應該啟用 `AutoStart` 或手動啟動應用程式。
- 本機安裝的 PySpark 版本必須與 Amazon EMR Serverless 應用程式上的 Apache Spark 版本相符（適用於 `3.5.6emr-7.13.0`）。版本不相符會導致 `ImportError` 或未預期的行為。
- Spark Connect 工作階段不支援透過 Lake Formation 進行精細存取控制。
- 使用 Spark Connect 的互動式工作階段不支援受信任身分傳播。
- 使用 Spark Connect 的互動式工作階段不支援 EMR Serverless 上的無伺服器儲存。
- 使用 Spark Connect 無需額外費用。您只需為工作階段期間消耗的 EMR Serverless 運算資源（vCPU、記憶體和儲存）付費。
- Spark 組態由 EMR Serverless `spark.connect.grpc.binding.address` 保留，使用者無法覆寫。
- 您在本機安裝的 PySpark 套件必須符合 EMR Serverless 應用程式的 Spark 版本。版本不相符會導致連線錯誤。Python UDFs (`@udf`、`spark.udf.register`) 也需要本機 Python 次要版本以符合工作者，或使用失敗 `PYTHON_VERSION_MISMATCH`。內建 SQL 函數和 DataFrame 操作不需要 Python 版本比對。
- 若要使用傳遞 Spark 組態 `start-session`，請在 `--configuration-overrides` 參數 `runtimeConfiguration` 中的下進行設定。`start-job-run` API 會 `applicationConfiguration` 改用。

透過 EMR Studio 使用 EMR Serverless 執行互動式工作負載

使用 EMR Serverless 互動式應用程式，使用 EMR Studio 中託管的筆記本，為 Spark 搭配 EMR Serverless 執行互動式工作負載。

概觀

互動式應用程式是已啟用互動式功能的 EMR Serverless 應用程式。使用 Amazon EMR Serverless 互動式應用程式，您可以使用 Amazon EMR Studio 中管理的 Jupyter 筆記本來執行互動式工作負載。這有助於資料工程師、資料科學家和資料分析師使用 EMR Studio 對 Amazon S3 和 Amazon DynamoDB 等資料存放區中的資料集執行互動式分析。

EMR Serverless 中互動式應用程式的使用案例包括下列項目：

- 資料工程師使用 EMR Studio 中的 IDE 體驗來建立 ETL 指令碼。指令碼從內部部署擷取資料、轉換資料進行分析，並將資料存放在 Amazon S3 中。
- 資料科學家使用筆記本來探索資料集，並訓練機器學習 (ML) 模型來偵測資料集中的異常。
- 資料分析師會探索資料集並建立指令碼，以產生每日報告來更新業務儀表板等應用程式。

先決條件

若要搭配 EMR Serverless 使用互動式工作負載，請滿足下列需求：

- Amazon EMR 6.14.0 及更高版本支援 EMR Serverless 互動式應用程式。
- 若要存取您的互動式應用程式、執行您提交的工作負載，以及從 EMR Studio 執行互動式筆記本，您需要特定的許可和角色。如需詳細資訊，請參閱 [互動式工作負載的必要許可](#)。

互動式工作負載的必要許可

除了[存取 EMR Serverless 所需的基本許可](#)之外，請為您的 IAM 身分或角色設定其他許可：

存取您的互動式應用程式

設定 EMR Studio 的使用者和工作區許可。如需詳細資訊，請參閱《Amazon [EMR 管理指南](#)》中的[設定 EMR Studio 使用者許可](#)。

執行您使用 EMR Serverless 提交的工作負載

設定任務執行時間角色。如需詳細資訊，請參閱 [建立任務執行期角色](#)。

從 EMR Studio 執行互動式筆記本

將下列額外許可新增至 Studio 使用者的 IAM 政策：

- **emr-serverless:AccessInteractiveEndpoints** - 准許存取並連線至您指定為的互動式應用程式Resource。從 EMR Studio 工作區連接到 EMR Serverless 應用程式需要此許可。
- **iam:PassRole** - 准許存取您在連接到應用程式時計劃使用的 IAM 執行角色。從 EMR Studio 工作區連接到 EMR Serverless 應用程式需要適當的PassRole許可。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessInteractiveAccess",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:AccessInteractiveEndpoints"
      ],
      "Resource": [
        "arn:aws:emr-serverless:*:123456789012:/applications/*"
      ]
    },
    {
      "Sid": "EMRServerlessRuntimeRoleAccess",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::123456789012:role/EMRServerlessInteractiveRole"
      ],
      "Condition": {
        "StringLike": {
          "iam:PassedToService": "emr-serverless.amazonaws.com"
        }
      }
    }
  ]
}
```

}

設定互動式應用程式

使用下列高階步驟，透過 中的 Amazon EMR Studio 建立具有互動式功能的 EMR Serverless 應用程式 AWS 管理主控台。

1. 遵循 中的步驟 [Amazon EMR Serverless 入門](#) 來建立應用程式。
2. 然後，從 EMR Studio 啟動工作區，並連接至 EMR Serverless 應用程式做為運算選項。如需詳細資訊，請參閱 [EMR Serverless 入門](#) 文件步驟 2 中的互動式工作負載索引標籤。

當您將應用程式連接到 Studio 工作區時，如果應用程式尚未執行，應用程式會自動啟動觸發。您也可以將應用程式連接至工作區之前，預先啟動應用程式並保持就緒狀態。

互動式應用程式的考量事項

- Amazon EMR 6.14.0 及更高版本支援 EMR Serverless 互動式應用程式。
- EMR Studio 是唯一與 EMR Serverless 互動式應用程式整合的用戶端。EMR Serverless 互動式應用程式不支援下列 EMR Studio 功能：工作區協同合作、SQL Explorer 和筆記本的程式設計執行。
- 只有 Spark 引擎支援互動式應用程式。
- 互動式應用程式支援 Python 3、PySpark 和 Spark Scala 核心。
- 您可以在單一互動式應用程式上執行最多 25 個並行筆記本。
- 沒有端點或 API 界面支援具有互動式應用程式的自我託管 Jupyter 筆記本。
- 為了獲得最佳化的啟動體驗，建議您為驅動程式和執行器設定預先初始化的容量，並預先啟動應用程式。當您預先啟動應用程式時，請確保在您想要將其連接至工作區時已準備就緒。

```
aws emr-serverless start-application \  
--application-id your-application-id
```

- 根據預設，autoStopConfig 會針對應用程式啟用。這會在閒置 30 分鐘後關閉應用程式。您可以在 create-application 或 update-application 請求中變更此組態。
- 使用互動式應用程式時，建議您設定核心、驅動程式和執行器的預先初始化容量，以執行筆記本。每個 Spark 互動式工作階段都需要一個核心和一個驅動程式，因此 EMR Serverless 會為每個預先初始化的驅動程式維護一個預先初始化的核心工作者。根據預設，EMR Serverless 在整個應用程式中維

護一個核心工作者的預先初始化容量，即使您未為驅動程式指定任何預先初始化容量。每個核心工作者使用 4 個 vCPU 和 16 GB 的記憶體。如需目前定價資訊，請參閱 [Amazon EMR 定價](#) 頁面。

- 您的 中必須有足夠的 vCPU 服務配額 AWS 帳戶，才能執行互動式工作負載。如果您不執行已啟用 Lake Formation 的工作負載，我們建議至少 24 個 vCPU。如果您這麼做，我們建議至少使用 28 個 vCPU。
- 如果核心閒置超過 60 分鐘，EMR Serverless 會自動從筆記本終止核心。EMR Serverless 會從筆記本工作階段期間完成的最後一個活動計算核心閒置時間。您目前無法修改核心閒置逾時設定。
- 若要使用互動式工作負載啟用 Lake Formation，請在建立 EMR Serverless 應用程式時，在 runtime-configuration 物件的 spark-defaults 分類 spark.emr-serverless.lakeformation.enabledtrue 下將組態設定為 <https://docs.aws.amazon.com/emr/latest/EMR-Serverless-UserGuide/getting-started.html> 若要進一步了解，請參閱在 [Amazon EMR 中啟用 Lake Formation](#)。

透過 Apache Livy 端點使用 EMR Serverless 執行互動式工作負載

使用 Amazon EMR 6.14.0 版和更新版本，在建立 EMR Serverless 應用程式時建立和啟用 Apache Livy 端點，並透過自我託管筆記本或使用自訂用戶端執行互動式工作負載。Apache Livy 端點提供下列優點：

- 您可以透過 Jupyter 筆記本安全地連線至 Apache Livy 端點，並使用 Apache Livy 的 REST 介面管理 Apache Spark 工作負載。
- 針對使用 Apache Spark 工作負載資料的互動 Web 應用程式，使用 Apache Livy REST API 操作。

先決條件

若要搭配 EMR Serverless 使用 Apache Livy 端點，請滿足下列需求：

- 完成 [Amazon EMR Serverless 入門](#) 中的步驟。
- 若要透過 Apache Livy 端點執行互動式工作負載，您需要特定許可和角色。如需詳細資訊，請參閱 [互動式工作負載的必要許可](#)。

所需的許可

除了存取 EMR Serverless 所需的許可之外，也請將下列許可新增至您的 IAM 角色，以存取 Apache Livy 端點並執行應用程式：

- `emr-serverless:AccessLivyEndpoints` – 授予許可，以存取和連線至您指定為 `已啟用 Livy` 的應用程式 `Resource`。您需要此許可才能從 Apache Livy 端點執行可用的 REST API 操作。
- `iam:PassRole` – 授予在建立 Apache Livy 工作階段時存取 IAM 執行角色的許可。EMR Serverless 將使用此角色來執行您的工作負載。
- `emr-serverless:GetDashboardForJobRun` – 授予許可，以產生 Spark Live UI 和驅動程式日誌連結，並提供日誌的存取權，做為 Apache Livy 工作階段結果的一部分。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessInteractiveAccess",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:AccessLivyEndpoints"
      ],
      "Resource": [
        "arn:aws:emr-serverless:*:123456789012:/applications/*"
      ]
    },
    {
      "Sid": "EMRServerlessRuntimeRoleAccess",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::123456789012:role/EMRServerlessExecutionRole"
      ],
      "Condition": {
        "StringLike": {
          "iam:PassedToService": "emr-serverless.amazonaws.com"
        }
      }
    },
    {
      "Sid": "EMRServerlessDashboardAccess",
      "Effect": "Allow",
```

```
"Action": [  
  "emr-serverless:GetDashboardForJobRun"  
],  
"Resource": [  
  "arn:aws:emr-serverless:*:123456789012:/applications/*"  
]  
}  
]  
}
```

開始使用

若要建立啟用 Apache Livy 的應用程式並執行它，請遵循下列步驟。

1. 若要建立啟用 Apache Livy 的應用程式，請執行下列命令。

```
aws emr-serverless create-application \  
--name my-application-name \  
--type 'application-type' \  
--release-label <Amazon EMR-release-version>  
--interactive-configuration '{"livyEndpointEnabled": true}'
```

2. EMR Serverless 建立應用程式後，請啟動應用程式，讓 Apache Livy 端點可用。

```
aws emr-serverless start-application \  
--application-id application-id
```

使用下列命令來檢查應用程式的狀態。一旦狀態變成 STARTED，請存取 Apache Livy 端點。

```
aws emr-serverless get-application \  
--region <AWS_REGION> --application-id >application_id>
```

3. 使用下列 URL 存取端點：

```
https://_application-id_.livy.emr-serverless-  
services._<AWS_REGION>_.amazonaws.com
```

一旦端點準備就緒，請根據您的使用案例提交工作負載。您必須使用 [SIGv4 通訊協定](#) 簽署每個對端點的請求，並傳入授權標頭。您可以使用下列方法來執行工作負載：

- HTTP 用戶端 – 使用自訂 HTTP 用戶端提交您的 Apache Livy 端點 API 操作。
- Sparkmagic 核心 – 在本機執行 Sparkmagic 核心，並使用 Jupyter 筆記本提交互動式查詢。

HTTP 用戶端

若要建立 Apache Livy 工作階段，請在請求內文的 `conf` 參數 `emr-serverless.session.executionRoleArn` 中提交。下列範例是範例 `POST /sessions` 請求。

```
{
  "kind": "pyspark",
  "heartbeatTimeoutInSecond": 60,
  "conf": {
    "emr-serverless.session.executionRoleArn": "<executionRoleArn>"
  }
}
```

下表說明所有可用的 Apache Livy API 操作。

API 操作	說明
GET /工作階段	傳回所有作用中互動式工作階段的清單。
POST/工作階段	透過 spark 或 pyspark 建立新的互動式工作階段。
GET /sessions/<sessionId >	傳回工作階段資訊。
GET /sessions/<sessionId >/state	傳回工作階段的狀態。
DELETE /sessions/<sessionId >	停止和刪除工作階段。
GET /sessions/<sessionId >/statements	傳回工作階段中的所有陳述式。
POST /sessions/<sessionId >/statements	在工作階段中執行陳述式。
GET /sessions/<sessionId >/statements/<statementId >	傳回工作階段中指定陳述式的詳細資訊。
POST /sessions/<sessionId >/statements/<statementId >/cancel	取消此工作階段中指定的陳述式。

將請求傳送至 Apache Livy 端點

您也可以從 HTTP 用戶端將請求直接傳送至 Apache Livy 端點。這樣做可讓您遠端執行筆記本外使用案例的程式碼。

開始傳送請求至端點之前，請確定您已安裝下列程式庫：

```
pip3 install botocore awscli requests
```

以下是將 HTTP 請求直接傳送到端點的範例 Python 指令碼：

```
from botocore import crt
import requests
from botocore.awsrequest import AWSRequest
from botocore.credentials import Credentials
import botocore.session
import json, pprint, textwrap

endpoint = 'https://<application_id>.livy.emr-serverless-
services.<AWS_REGION>.amazonaws.com'
headers = {'Content-Type': 'application/json'}

session = botocore.session.Session()
signer = crt.auth.CrtS3SigV4Auth(session.get_credentials(), 'emr-serverless',
    '<AWS_REGION>')

### Create session request

data = {'kind': 'pyspark', 'heartbeatTimeoutInSecond': 60, 'conf': { 'emr-
serverless.session.executionRoleArn': 'arn:aws:iam::123456789012:role/role1'}}

request = AWSRequest(method='POST', url=endpoint + "/sessions", data=json.dumps(data),
    headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r = requests.post(prepped.url, headers=prepped.headers, data=json.dumps(data))
```

```
pprint.pprint(r.json())

### List Sessions Request

request = AWSRequest(method='GET', url=endpoint + "/sessions", headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r2 = requests.get(prepped.url, headers=prepped.headers)
pprint.pprint(r2.json())

### Get session state

session_url = endpoint + r.headers['location']

request = AWSRequest(method='GET', url=session_url, headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r3 = requests.get(prepped.url, headers=prepped.headers)

pprint.pprint(r3.json())

### Submit Statement

data = {
    'code': "1 + 1"
}

statements_url = endpoint + r.headers['location'] + "/statements"

request = AWSRequest(method='POST', url=statements_url, data=json.dumps(data),
headers=headers)
```

```
request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r4 = requests.post(prepped.url, headers=prepped.headers, data=json.dumps(data))

pprint.pprint(r4.json())

### Check statements results

specific_statement_url = endpoint + r4.headers['location']

request = AWSRequest(method='GET', url=specific_statement_url, headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r5 = requests.get(prepped.url, headers=prepped.headers)

pprint.pprint(r5.json())

### Delete session

session_url = endpoint + r.headers['location']

request = AWSRequest(method='DELETE', url=session_url, headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r6 = requests.delete(prepped.url, headers=prepped.headers)
```

```
pprint.pprint(r6.json())
```

Sparkmagic 核心

安裝 sparkmagic 之前，請確定您已在要安裝 sparkmagic 的執行個體中設定 AWS 登入資料

1. 依照安裝 [步驟安裝](#) sparkmagic。請注意，您只執行前四個步驟。
2. Sparkmagic 核心支援自訂驗證程式，因此您可以將驗證程式與 Sparkmagic 核心整合，讓每個請求都經過 SIGv4 簽署。
3. 安裝 EMR Serverless 自訂驗證器。

```
pip install emr-serverless-customauth
```

4. 現在提供自訂驗證器的路徑，以及 Sparkmagic 組態 json 檔案中的 Apache Livy 端點 URL。使用下列命令開啟組態檔案。

```
vim ~/.sparkmagic/config.json
```

以下是範例 config.json 檔案。

```
{
  "kernel_python_credentials" : {
    "username": "",
    "password": "",
    "url": "https://<application-id>.livy.emr-serverless-
services.<AWS_REGION>.amazonaws.com",
    "auth": "Custom_Auth"
  },

  "kernel_scala_credentials" : {
    "username": "",
    "password": "",
    "url": "https://<application-id>.livy.emr-serverless-
services.<AWS_REGION>.amazonaws.com",
    "auth": "Custom_Auth"
  },
  "authenticators": {
    "None": "sparkmagic.auth.customauth.Authenticator",
    "Basic_Access": "sparkmagic.auth.basic.Basic",
    "Custom_Auth":
    "emr_serverless_customauth.customauthenticator.EMRServerlessCustomSigV4Signer"
```

```

    },
    "livy_session_startup_timeout_seconds": 600,
    "ignore_ssl_errors": false
  }
}

```

5. 啟動 Jupyter 實驗室。它應該使用您在最後一個步驟中設定的自訂身分驗證。
6. 然後，您可以執行下列筆記本命令和程式碼以開始使用。

```
%info //Returns the information about the current sessions.
```

```

%%configure -f //Configure information specific to a session. We supply
executionRoleArn in this example. Change it for your use case.
{
  "driverMemory": "4g",
  "conf": {
    "emr-serverless.session.executionRoleArn":
"arn:aws:iam::123456789012:role/JobExecutionRole"
  }
}

```

```
<your code>//Run your code to start the session
```

在內部，每個指令會透過設定的 Apache Livy 端點 URL 呼叫每個 Apache Livy API 操作。然後，您可以根據您的使用案例撰寫指示。

考量事項

透過 Apache Livy 端點執行互動式工作負載時，請考慮下列事項。

- EMR Serverless 使用發起人主體維持工作階段層級隔離。建立工作階段的呼叫者主體是唯一可以存取該工作階段的主體。如需更精細的隔離，請在取得登入資料時設定來源身分。在此情況下，EMR Serverless 會根據發起人主體和來源身分強制執行工作階段層級隔離。如需來源身分的詳細資訊，請參閱使用[擔任的角色所採取的監控和控制動作](#)。
- EMR Serverless 6.14.0 版及更新版本支援 Apache Livy 端點。
- 僅 Apache Spark 引擎支援 Apache Livy 端點。
- Apache Livy 端點支援 Scala Spark 和 PySpark。

- 根據預設，`autoStopConfig` 會在您的應用程式中啟用。這表示應用程式在閒置 15 分鐘後關閉。您可以在 `create-application` 或 `update-application` 請求中變更此組態。
- 您可以在啟用 Apache Livy 端點的應用程式上執行最多 25 個並行工作階段。
- 為了獲得最佳的啟動體驗，我們建議您為驅動程式和執行器設定預先初始化的容量。
- 您必須先手動啟動應用程式，才能連線至 Apache Livy 端點。
- 您的中必須有足夠的 vCPU 服務配額 AWS 帳戶，才能使用 Apache Livy 端點執行互動式工作負載。我們建議至少 24 個 vCPU。
- 預設 Apache Livy 工作階段逾時為 1 小時。如果您超過一小時未執行陳述式，則 Apache Livy 會刪除工作階段，並釋出驅動程式和執行器。從版本 `emr-7.8.0` 開始，此值可以透過將 `ttl` 參數指定為 Livy `/sessions` POST 請求的一部分來設定，例如 2h (小時)、120m (分鐘)、7200s (秒)、7200000ms (毫秒)。

Note

此組態無法在 `emr-7.8.0` 之前變更。以下是 POST `/sessions` 請求內文的範例。

```
{
  "kind": "pyspark",
  "heartbeatTimeoutInSeconds": 60,
  "conf": {
    "emr-serverless.session.executionRoleArn": "executionRoleArn"
  },
  "ttl": "2h"
}
```

- 從啟用 Lake Formation 的精細存取控制應用程式的 Amazon EMR 發行版本 `emr-7.8.0` 開始，每個工作階段都可以停用設定。如需為 EMR Serverless 應用程式啟用精細存取控制的詳細資訊，請參閱[精細存取控制的方法](#)。

Note

當工作階段尚未為應用程式啟用 Lake Formation 時，就無法為其啟用。以下是 POST `/sessions` 請求內文的範例。

```
{
```

```
"kind": "pyspark",
"heartbeatTimeoutInSeconds": 60,
"conf": {
  "emr-serverless.session.executionRoleArn": "executionRoleArn"
},
"spark.emr-serverless.lakeformation.enabled" : "false"
}
```

- 只有作用中工作階段才能與 Apache Livy 端點互動。一旦工作階段完成、取消或終止，您就無法透過 Apache Livy 端點存取它。

日誌記錄和監控

監控是維護 EMR Serverless 應用程式和任務可靠性、可用性和效能的重要部分。您應該從 EMR Serverless 解決方案的所有部分收集監控資料，以便在發生多點故障時更輕鬆地進行偵錯。

主題

- [儲存日誌](#)
- [輪換日誌](#)
- [加密日誌](#)
- [設定 Amazon EMR Serverless 的 Apache Log4j2 屬性](#)
- [監控 EMR Serverless](#)
- [使用 自動化 EMR Serverless Amazon EventBridge](#)

儲存日誌

若要在 EMR Serverless 上監控任務進度並疑難排解任務失敗，請選擇 EMR Serverless 如何存放和提供應用程式日誌。當您提交任務執行時，請指定 受管儲存、Amazon S3 和 Amazon CloudWatch 做為您的記錄選項。

使用 CloudWatch，指定您要使用的日誌類型和日誌位置，或接受預設類型和位置。如需 CloudWatch 日誌的詳細資訊，請參閱 [the section called “Amazon CloudWatch”](#)。使用受管儲存和 S3 記錄，下表列出如果您選擇 [受管儲存](#)、[Amazon S3 儲存貯體](#) 或兩者，您可以預期的日誌位置和 UI 可用性。

選項	事件日誌	容器日誌	應用程式 UI
受管儲存	存放在受管儲存體	存放在受管儲存體	支援
受管儲存和 S3 儲存貯體	存放在這兩個位置	存放在 S3 儲存貯體	支援
Amazon S3 儲存貯體	存放在 S3 儲存貯體	存放在 S3 儲存貯體	不支援 ¹

¹ 建議您保持選取受管儲存選項。否則，您無法使用內建應用程式 UIs。

使用受管儲存體記錄 EMR Serverless

根據預設，EMR Serverless 會將應用程式日誌安全地存放在 Amazon EMR 受管儲存體中，最長可達 30 天。

Note

如果您關閉預設選項，Amazon EMR 無法代表您疑難排解任務。範例：您無法從 EMR Serverless Console 存取 Spark-UI。

若要從 EMR Studio 關閉此選項，請在提交任務頁面的其他設定區段取消選取允許 AWS 保留日誌 30 天核取方塊。

若要從 關閉此選項 AWS CLI，請在提交任務執行時使用 `managedPersistenceMonitoringConfiguration` 組態。

```
{
  "monitoringConfiguration": {
    "managedPersistenceMonitoringConfiguration": {
      "enabled": false
    }
  }
}
```

如果您的 EMR Serverless 應用程式位於具有 Amazon S3 VPC 端點的私有子網路中，而且您連接端點政策來控制存取，請新增下列許可，讓 EMR Serverless 存放和提供應用程式日誌。Resource 將取代之為[存取 Amazon S3 之私有子網路的範例政策](#)中可用區域資料表中的 AppInfo 儲存貯體。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessManagedLogging",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
```

```

    "s3:PutObjectAcl"
  ],
  "Resource": [
    "arn:aws:s3:::prod.us-east-1.appinfo.src",
    "arn:aws:s3:::prod.us-east-1.appinfo.src/*"
  ],
  "Condition": {
    "StringEquals": {
      "aws:PrincipalServiceName": "emr-serverless.amazonaws.com",
      "aws:SourceVpc": "vpc-12345678"
    }
  }
}
]
}

```

此外，請使用 `aws:SourceVpc` 條件金鑰，以確保請求通過 VPC 端點連接的 VPC。

使用 Amazon S3 儲存貯體記錄 EMR Serverless

在您的任務可以將日誌資料傳送到 Amazon S3 之前，請在任務執行時間角色的許可政策中包含下列許可。`amzn-s3-demo-logging-bucket` 將取代為記錄儲存貯體的名稱。

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ],
      "Sid": "AllowS3Putobject"
    }
  ]
}

```

若要設定 Amazon S3 儲存貯體來存放來自的日誌 AWS CLI，請在開始任務執行時使用 `s3MonitoringConfiguration` 組態。若要這樣做，請在組態 `--configuration-overrides` 中提供下列項目。

```
{
  "monitoringConfiguration": {
    "s3MonitoringConfiguration": {
      "logUri": "s3://amzn-s3-demo-logging-bucket/logs/"
    }
  }
}
```

對於未啟用重試的批次任務，EMR Serverless 會將日誌傳送至下列路徑：

```
'/applications/<applicationId>/jobs/<jobId>'
```

EMR Serverless 會將 Spark 驅動程式日誌儲存在下列路徑中

```
'/applications/<applicationId>/jobs/<jobId>/SPARK_DRIVER/'
```

Spark 執行器日誌由 EMR Serverless 存放在下列路徑中

```
'/applications/<applicationId>/jobs/<jobId>/SPARK_EXECUTOR/<EXECUTOR-ID>'
```

<EXECUTOR-ID> 是整數。

EMR Serverless 7.1.0 版和更新版本支援串流任務和批次任務的重試嘗試。如果您在啟用重試的情況下執行任務，EMR Serverless 會自動將嘗試次數新增至日誌路徑字首，以便更好地區分和追蹤日誌。

```
'/applications/<applicationId>/jobs/<jobId>/attempts/<attemptNumber>/'
```

使用 Amazon CloudWatch 記錄 EMR Serverless

當您將任務提交至 EMR Serverless 應用程式時，請選擇 Amazon CloudWatch 做為儲存應用程式日誌的選項。這可讓您使用 CloudWatch Logs Insights 和 Live Tail 等 CloudWatch Logs 分析功能。您也可以將日誌從 CloudWatch 串流到 OpenSearch 等其他系統，以進行進一步分析。

EMR Serverless 提供驅動程式日誌的即時記錄。您可以使用 CloudWatch 即時結尾功能或透過 CloudWatch CLI 結尾命令即時存取日誌。

預設會停用 EMR Serverless 的 CloudWatch 記錄。若要啟用此功能，請使用中的組態[AWS CLI](#)。

Note

Amazon CloudWatch 會即時發佈日誌，因此從工作者產生更多資源。如果您選擇低工作者容量，對任務執行時間的影響可能會增加。如果您啟用 CloudWatch 記錄，我們建議您選擇更大的工作者容量。如果的交易每秒 (TPS) 速率太低，日誌發佈也可能會調節 PutLogEvents。CloudWatch 限流組態適用於所有服務，包括 EMR Serverless。如需詳細資訊，請參閱[如何在 CloudWatch 日誌中判斷限流？](#) AWS re:post 上的。

使用 CloudWatch 記錄所需的許可

在您的任務可以將日誌資料傳送至 Amazon CloudWatch 之前，請在任務執行時間角色的許可政策中包含下列許可。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:*:123456789012:*"
      ],
      "Sid": "AllowLOGSDescribeLogGroups"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:*:123456789012:log-group:my-log-group-name:*"
      ]
    }
  ]
}
```

```

    ],
    "Sid": "AllowLOGSPutlogevents"
  }
]
}

```

AWS CLI

若要設定 Amazon CloudWatch 從存放 EMR Serverless 的日誌 AWS CLI，請在開始任務執行時使用 `cloudWatchLoggingConfiguration` 組態。若要這樣做，請提供下列組態覆寫。或者，您也可以提供日誌群組名稱、日誌串流字首名稱、日誌類型和加密金鑰 ARN。

如果您未指定選用值，則 CloudWatch 會使用預設日誌串流，將日誌發佈至 `/aws/emr-serverless` 預設日誌群組 `/applications/applicationId/jobs/jobId/worker-type`。

EMR Serverless 7.1.0 版和更新版本支援串流任務和批次任務的重試嘗試。如果您啟用任務的重試，EMR Serverless 會自動將嘗試次數新增至日誌路徑字首，以便您更清楚區分和追蹤日誌。

```
'/applications/<applicationId>/jobs/<jobId>/attempts/<attemptNumber>/worker-type'
```

以下示範使用 EMR Serverless 的預設設定開啟 Amazon CloudWatch 記錄所需的最低組態：

```

{
  "monitoringConfiguration": {
    "cloudWatchLoggingConfiguration": {
      "enabled": true
    }
  }
}

```

下列範例顯示您為 EMR Serverless 開啟 Amazon CloudWatch 記錄時指定的所有必要和選用組態。支援的 `logTypes` 值也會列在下列範例中。

```

{
  "monitoringConfiguration": {
    "cloudWatchLoggingConfiguration": {
      "enabled": true, // Required
      "logGroupName": "Example_logGroup", // Optional
      "logStreamNamePrefix": "Example_logStream", // Optional
    }
  }
}

```

```
    "encryptionKeyArn": "key-arn", // Optional
    "logTypes": {
      "SPARK_DRIVER": ["stdout", "stderr"] //List of values
    }
  }
}
```

根據預設，EMR Serverless 只會將驅動程式 stdout 和 stderr 日誌發佈至 CloudWatch。如果您想要其他日誌，請使用 logTypes 欄位指定容器角色和對應的日誌類型。

下列清單顯示為 logTypes 組態指定的支援工作者類型：

Spark

- SPARK_DRIVER : ["STDERR", "STDOUT"]
- SPARK_EXECUTOR : ["STDERR", "STDOUT"]

Hive

- HIVE_DRIVER : ["STDERR", "STDOUT", "HIVE_LOG", "TEZ_AM"]
- TEZ_TASK : ["STDERR", "STDOUT", "SYSTEM_LOGS"]

輪換日誌

Amazon EMR Serverless 可以輪換 Spark 應用程式日誌和事件日誌。日誌輪換有助於解決長時間執行任務的問題，產生可以佔用所有磁碟空間的大型日誌檔案。輪換日誌可協助您節省磁碟儲存體並減少任務失敗的數量，因為您磁碟上沒有剩餘的空間。

日誌輪換預設為啟用，且僅適用於 Spark 任務。

Spark 事件日誌

Note

Spark 事件日誌輪換適用於所有 Amazon EMR 發行標籤。

EMR Serverless 不會產生單一事件日誌檔案，而是定期輪換事件日誌，並移除較舊的事件日誌檔案。輪換日誌不會影響上傳至 S3 儲存貯體的日誌。

Spark 應用程式日誌

Note

Spark 應用程式日誌輪換適用於所有 Amazon EMR 發行標籤。

EMR Serverless 也會輪換驅動程式和執行器的 Spark 應用程式日誌，例如 `stdout` 和 `stderr` 檔案。您可以使用 Spark 歷史記錄伺服器 and 即時 UI 連結，選擇 Studio 中的日誌連結來存取最新的日誌檔案。日誌檔案是最新日誌的截斷版本。若要參考較舊的輪換日誌，請在儲存日誌時指定 Amazon S3 位置。如需詳細資訊，請參閱 [使用 Amazon S3 儲存貯體記錄 EMR Serverless](#)。

您可以在下列位置找到最新的日誌檔案。EMR Serverless 每 15 秒重新整理一次檔案。這些檔案的範圍可以從 0 MB 到 128 MB。

```
<example-S3-logUri>/applications/<application-id>/jobs/<job-id>/SPARK_DRIVER/stderr.gz
```

下列位置包含較舊的輪換檔案。每個檔案為 128 MB。

```
<example-S3-logUri>/applications/<application-id>/jobs/<job-id>/SPARK_DRIVER/archived/  
stderr_<index>.gz
```

相同的行為也適用於 Spark 執行器。此變更僅適用於 S3 記錄。日誌輪換不會對上傳至 Amazon CloudWatch 的日誌串流引入任何變更。

EMR Serverless 7.1.0 版和更新版本支援串流和批次任務的重試嘗試。如果您已啟用對任務的重試嘗試，EMR Serverless 會將字首新增至此類任務的日誌路徑，以便您可以更好地追蹤和區分日誌。此路徑包含所有輪換的日誌。

```
 '/applications/<applicationId>/jobs/<jobId>/attempts/<attemptNumber>/'.
```

加密日誌

使用受管儲存體加密 EMR Serverless 日誌

若要使用您自己的 KMS 金鑰加密受管儲存體中的日誌，請在提交任務執行時使用 `managedPersistenceMonitoringConfiguration` 組態。

```
{
  "monitoringConfiguration": {
    "managedPersistenceMonitoringConfiguration" : {
      "encryptionKeyArn": "key-arn"
    }
  }
}
```

使用 Amazon S3 儲存貯體加密 EMR Serverless 日誌

若要使用您自己的 KMS 金鑰加密 Amazon S3 儲存貯體中的日誌，請在提交任務執行時使用 `s3MonitoringConfiguration` 組態。

```
{
  "monitoringConfiguration": {
    "s3MonitoringConfiguration": {
      "logUri": "s3://amzn-s3-demo-logging-bucket/logs/",
      "encryptionKeyArn": "key-arn"
    }
  }
}
```

使用 Amazon CloudWatch 加密 EMR Serverless 日誌

若要使用您自己的 KMS 金鑰加密 Amazon CloudWatch 中的日誌，請在提交任務執行時使用 `cloudWatchLoggingConfiguration` 組態。

```
{
  "monitoringConfiguration": {
    "cloudWatchLoggingConfiguration": {
      "enabled": true,
      "encryptionKeyArn": "key-arn"
    }
  }
}
```

日誌加密的必要許可

本節內容

- [必要的使用者許可](#)

- [Amazon S3 和受管儲存體的加密金鑰許可](#)
- [Amazon CloudWatch 的加密金鑰許可](#)

必要的使用者許可

提交任務或檢視日誌或應用程式 UIs 的使用者必須具有使用金鑰的許可。您可以在 KMS 金鑰政策或使用者、群組或角色的 IAM 政策中指定許可。如果提交任務的使用者缺少 KMS 金鑰許可，EMR Serverless 會拒絕任務執行提交。

範例金鑰政策

下列金鑰政策提供 `kms:GenerateDataKey` 和 `kms:Decrypt` 的許可：

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/user-name"
  },
  "Action": [
    "kms:GenerateDataKey",
    "kms:Decrypt"
  ],
  "Resource": "*"
}
```

IAM 政策範例

下列 IAM 政策提供 `kms:GenerateDataKey` 和 `kms:Decrypt` 的許可：

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
    }
  ]
}
```

```

    "Resource": [
      "arn:aws:kms:*:123456789012:key/12345678-1234-1234-1234-123456789012"
    ],
    "Sid": "AllowKMSGeneratedakey"
  }
]
}

```

若要啟動 Spark 或 Tez UI，請授予您的使用者、群組或角色存取 `emr-serverless:GetDashboardForJobRun` API 的許可，如下所示：

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:GetDashboardForJobRun"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowEMRSERVERLESSGetdashboardforjobrun"
    }
  ]
}

```

Amazon S3 和受管儲存體的加密金鑰許可

當您在受管儲存體或 S3 儲存貯體中使用自己的加密金鑰加密日誌時，請設定 KMS 金鑰許可，如下所示。

`emr-serverless.amazonaws.com` 委託人必須在 KMS 金鑰的政策中具有下列許可：

```

{
  "Effect": "Allow",
  "Principal": {
    "Service": "emr-serverless.amazonaws.com"
  }
}

```

```

    },
    "Action": [
      "kms:Decrypt",
      "kms:GenerateDataKey"
    ],
    "Resource": "*"
  "Condition": {
    "StringLike": {
      "aws:SourceArn": "arn:aws:emr-serverless:region:aws-account-id:/
applications/application-id"
    }
  }
}

```

作為安全最佳實務，我們建議您將 `aws:SourceArn` 條件金鑰新增至 KMS 金鑰政策。IAM 全域條件金鑰 `aws:SourceArn` 有助於確保 EMR Serverless 僅針對應用程式 ARN 使用 KMS 金鑰。

任務執行期角色在其 IAM 政策中必須具有下列許可：

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:kms:*:123456789012:key/12345678-1234-1234-1234-123456789012"
      ],
      "Sid": "AllowKMSGeneratedatakey"
    }
  ]
}

```

Amazon CloudWatch 的加密金鑰許可

若要將 KMS 金鑰 ARN 與您的日誌群組建立關聯，請針對任務執行期角色使用下列 IAM 政策。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:AssociateKmsKey"
      ],
      "Resource": [
        "arn:aws:logs:*:123456789012:log-group:my-log-group-name:*"
      ],
      "Sid": "AllowLOGSAssociatekmskey"
    }
  ]
}
```

設定 KMS 金鑰政策，將 KMS 許可授予 Amazon CloudWatch：

JSON

```
{
  "Version": "2012-10-17",
  "Id": "key-default-1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "ArnLike": {
          "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:*:123456789012:*"
        }
      },
      "Sid": "AllowKMSDecrypt"
    }
  ]
}
```

```
}  
]  
}
```

設定 Amazon EMR Serverless 的 Apache Log4j2 屬性

此頁面說明如何為位於的 EMR Serverless 任務設定自訂 [Apache Log4j 2.x 屬性](#)。StartJobRun 如果您想要在應用程式層級設定 Log4j 分類，請參閱 [EMR Serverless 的預設應用程式組態](#)。

設定 Amazon EMR Serverless 的 Spark Log4j2 屬性

透過 Amazon EMR 6.8.0 版和更新版本，您可以自訂 [Apache Log4j 2.x 屬性](#) 來指定精細的日誌組態。這可簡化 EMR Serverless 上 Spark 任務的疑難排解。若要設定這些屬性，請使用 `spark-driver-log4j2` 和 `spark-executor-log4j2` 分類。

主題

- [Spark 的 Log4j2 分類](#)
- [Spark 的 Log4j2 組態範例](#)
- [範例 Spark 任務中的 Log4j2](#)
- [Spark 的 Log4j2 考量事項](#)

Spark 的 Log4j2 分類

若要自訂 Spark 日誌組態，請使用下列分類搭配 [applicationConfiguration](#)。若要設定 Log4j 2.x 屬性，請使用下列 [properties](#)。

spark-driver-log4j2

此分類會為驅動程式設定 `log4j2.properties` 檔案中的值。

spark-executor-log4j2

此分類會設定執行器 `log4j2.properties` 檔案中的值。

Spark 的 Log4j2 組態範例

下列範例示範如何使用提交 Spark 任務 `applicationConfiguration`，以自訂 Spark 驅動程式和執行器的 Log4j2 組態。

若要在應用程式層級設定 Log4j 分類，而不是在您提交任務時設定，請參閱 [EMR Serverless 的預設應用程式組態](#)。

```
aws emr-serverless start-job-run \  
  --application-id application-id \  
  --execution-role-arn job-role-arn \  
  --job-driver '{  
    "sparkSubmit": {  
      "entryPoint": "/usr/lib/spark/examples/jars/spark-examples.jar",  
      "entryPointArguments": ["1"],  
      "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi --conf  
spark.executor.cores=4 --conf spark.executor.memory=20g --conf spark.driver.cores=4 --  
conf spark.driver.memory=8g --conf spark.executor.instances=1"  
    }  
  }'  
  --configuration-overrides '{  
    "applicationConfiguration": [  
      {  
        "classification": "spark-driver-log4j2",  
        "properties": {  
          "rootLogger.level": "error", // will only display Spark error logs  
          "logger.IdentifierForClass.name": "classpath for setting logger",  
          "logger.IdentifierForClass.level": "info"  
        }  
      },  
      {  
        "classification": "spark-executor-log4j2",  
        "properties": {  
          "rootLogger.level": "error", // will only display Spark error logs  
          "logger.IdentifierForClass.name": "classpath for setting logger",  
          "logger.IdentifierForClass.level": "info"  
        }  
      }  
    ]  
  }'
```

範例 Spark 任務中的 Log4j2

下列程式碼範例示範如何在初始化應用程式的自訂 Log4j2 組態時建立 Spark 應用程式。

Python

Example- 使用 Log4j2 搭配 Python 進行 Spark 任務

```
import os
import sys

from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession

app_name = "PySparkApp"
if __name__ == "__main__":
    spark = SparkSession\
        .builder\
        .appName(app_name)\
        .getOrCreate()

    spark.sparkContext._conf.getAll()
    sc = spark.sparkContext
    log4jLogger = sc._jvm.org.apache.log4j
    LOGGER = log4jLogger.LogManager.getLogger(app_name)

    LOGGER.info("pyspark script logger info")
    LOGGER.warn("pyspark script logger warn")
    LOGGER.error("pyspark script logger error")

    // your code here

    spark.stop()
```

若要在執行 Spark 任務時自訂驅動程式的 Log4j2，請使用下列組態：

```
{
  "classification": "spark-driver-log4j2",
  "properties": {
    "rootLogger.level": "error", // only display Spark error logs
    "logger.PySparkApp.level": "info",
    "logger.PySparkApp.name": "PySparkApp"
  }
}
```

Scala

Example- 使用 Log4j2 搭配 Scala 進行 Spark 任務

```
import org.apache.log4j.Logger
import org.apache.spark.sql.Session

object ExampleClass {
  def main(args: Array[String]): Unit = {
    val spark = SparkSession
      .builder
      .appName(this.getClass.getName)
      .getOrCreate()

    val logger = Logger.getLogger(this.getClass);
    logger.info("script logging info logs")
    logger.warn("script logging warn logs")
    logger.error("script logging error logs")

    // your code here
    spark.stop()
  }
}
```

若要在執行 Spark 任務時自訂驅動程式的 Log4j2，請使用下列組態：

```
{
  "classification": "spark-driver-log4j2",
  "properties": {
    "rootLogger.level": "error", // only display Spark error logs
    "logger.ExampleClass.level": "info",
    "logger.ExampleClass.name": "ExampleClass"
  }
}
```

Spark 的 Log4j2 考量事項

Spark 程序無法設定下列 Log4j2.x 屬性：

- `rootLogger.appenderRef.stdout.ref`
- `appender.console.type`

- `appender.console.name`
- `appender.console.target`
- `appender.console.layout.type`
- `appender.console.layout.pattern`

如需設定之 Log4j2.x 屬性的詳細資訊，請參閱 GitHub 上的 [log4j2.properties.template 檔案](#)。

監控 EMR Serverless

本節涵蓋監控 Amazon EMR Serverless 應用程式和任務的方式。

主題

- [監控 EMR Serverless 應用程式和任務](#)
- [使用 Amazon Managed Service for Prometheus 監控 Spark 指標](#)
- [EMR Serverless 用量指標](#)

監控 EMR Serverless 應用程式和任務

透過 EMR Serverless 的 Amazon CloudWatch 指標，您可以接收 1 分鐘的 CloudWatch 指標並存取 CloudWatch 儀表板，以存取 EMR Serverless 應用程式的 near-real-time 操作和效能。

EMR Serverless 每分鐘將指標傳送至 CloudWatch。EMR Serverless 會在應用程式層級以及任務、工作者類型和 capacity-allocation-type 層級發出這些指標。

若要開始使用，請使用 EMR Serverless [GitHub 儲存庫中提供的 EMR Serverless](#) CloudWatch 儀表板範本並進行部署。

Note

[EMR Serverless 互動式工作負載](#) 僅啟用應用程式層級監控，並具有新的工作者類型維度 `Spark_Kernel`。若要監控和偵錯互動式工作負載，請從 [EMR Studio 工作區](#) 中存取日誌和 Apache Spark UI。

監控指標

Important

我們正在重組指標顯示以新增 ApplicationName 和 JobName 做為維度。對於 7.10 版和更新版本，舊指標將不再更新。對於低於 7.10 的 EMR 版本，舊版指標仍然可用。

目前維度

下表說明 AWS/EMR Serverless 命名空間中可用的 EMR Serverless 維度。

EMR Serverless 指標的維度

維度	Description
ApplicationId	使用應用程式 ID 篩選 EMR Serverless 應用程式的所有指標。
ApplicationName	使用名稱篩選 EMR Serverless 應用程式的所有指標。如果未提供名稱，或包含非 ASCII 字元，則會發佈為【未指定】。
JobId	篩選任務執行 ID 的 EMR Serverless 的所有指標。
JobName	使用名稱篩選 EMR Serverless 任務執行的所有指標。如果未提供名稱，或包含非 ASCII 字元，則會發佈為【未指定】。
WorkerType	篩選指定工作者類型的所有指標。例如，您可以篩選 Spark 任務的 SPARK_DRIVER 和 SPARK_EXECUTORS。

維度	Description
CapacityAllocationType	篩選指定容量分配類型的所有指標。例如，您可以篩選 PreInitCapacity 的預先初始化容量和其他 OnDemandCapacity 項目。

應用程式層級監控

您可以使用 Amazon CloudWatch 指標監控 EMR Serverless 應用程式層級的容量使用量。您也可以設定單一顯示器來監控 CloudWatch 儀表板中的應用程式容量用量。

EMR Serverless 應用程式指標

指標	Description	單位	維度
MaxCPUAllowed	應用程式允許的 CPU 上限。	vCPU	ApplicationId , ApplicationName
MaxMemoryAllowed	應用程式允許的記憶體上限，以 GB 為單位。	GB (GB)	ApplicationId , ApplicationName
MaxStorageAllowed	應用程式的允許最大儲存體，以 GB 為單位。	GB (GB)	ApplicationId , ApplicationName
CPUAllocated	配置 vCPUs 總數。	vCPU	ApplicationId , ApplicationName , WorkerType , CapacityAllocationType
IdleWorkerCount	工作者閒置總數。	計數	ApplicationId , ApplicationName , WorkerType

指標	Description	單位	維度
			e ,CapacityAllocationType
MemoryAllocated	配置的總記憶體，以 GB 為單位。	GB (GB)	ApplicationId , ApplicationName , WorkerType ,CapacityAllocationType
PendingCreationWorkerCount	待建立的工作者總數。	計數	ApplicationId , ApplicationName , WorkerType ,CapacityAllocationType
RunningWorkerCount	應用程式正在使用的工作者總數。	計數	ApplicationId , ApplicationName , WorkerType ,CapacityAllocationType
StorageAllocated	配置的總磁碟儲存體，以 GB 為單位。	GB (GB)	ApplicationId , ApplicationName , WorkerType ,CapacityAllocationType
TotalWorkerCount	可用的工作者總數。	計數	ApplicationId , ApplicationName , WorkerType ,CapacityAllocationType

任務層級監控

Amazon EMR Serverless Amazon CloudWatch 每一分鐘會將下列任務層級指標傳送至 。您可以存取依任務執行狀態彙總任務執行的指標值。每個指標的單位都是計數。

EMR Serverless 任務層級指標

指標	Description	維度
SubmittedJobs	處於已提交狀態的任務數量。	ApplicationId , ApplicationName
PendingJobs	處於待定狀態的任務數量。	ApplicationId , ApplicationName
ScheduledJobs	處於排程狀態的任務數量。	ApplicationId , ApplicationName
RunningJobs	處於執行中狀態的任務數量。	ApplicationId , ApplicationName
SuccessJobs	處於成功狀態的任務數量。	ApplicationId , ApplicationName
FailedJobs	處於失敗狀態的任務數量。	ApplicationId , ApplicationName
CancellingJobs	處於取消狀態的任務數量。	ApplicationId , ApplicationName
CancelledJobs	處於已取消狀態的任務數量。	ApplicationId , ApplicationName

您可以使用引擎特定應用程式 UIs 監控引擎特定指標，以執行和完成 EMR Serverless 任務。當您存取執行中任務的 UI 時，即時應用程式 UI 會顯示即時更新。當您存取已完成任務的 UI 時，會顯示持久性應用程式 UI。

執行任務

對於執行中的 EMR Serverless 任務，請存取提供引擎特定指標的即時界面。您可以使用 Apache Spark UI 或 Hive Tez UI 來監控和偵錯任務。若要存取這些 UIs，請使用 EMR Studio 主控台或透過 請求安全 URL 端點 AWS Command Line Interface。

已完成的任務

對於您已完成的 EMR Serverless 任務，請使用 Spark 歷史記錄伺服器或持久性 Hive Tez UI 來存取 Spark 或 Hive 任務執行的任務詳細資訊、階段、任務和指標。若要存取這些 UIs，請使用 EMR Studio 主控台，或使用 請求安全 URL 端點 AWS Command Line Interface。

任務工作者層級監控

Amazon EMR Serverless 會將 AWS/EMR Serverless 命名空間和 Job Worker Metrics 指標群組中可用的下列任務工作者層級指標傳送至 Amazon CloudWatch。EMR Serverless 會在任務層級、工作者類型和 capacity-allocation-type 層級的任務執行期間，收集個別工作者的資料點。您可以使用 ApplicationId 做為維度，以監控屬於相同應用程式的多個任務。

Note

若要在 Amazon CloudWatch 主控台中檢視指標時，檢視 EMR Serverless 任務所使用的 CPU 和記憶體總數，請使用統計資料為總和，並使用期間為 1 分鐘。

EMR Serverless 任務工作者層級指標

指標	Description	單位	維度
WorkerCpuAllocated	任務執行中為工作者配置的 vCPU 核心總數。	vCPU	JobId、JobName、ApplicationId、ApplicationName、WorkerType 和 CapacityAllocationType
WorkerCpuUsed	工作執行中工作者使用的 vCPU 核心總數。	vCPU	JobId、JobName、ApplicationId、ApplicationName、WorkerType 和

指標	Description	單位	維度
			CapacityAllocationType
WorkerMemoryAllocated	任務執行中為工作者配置的總記憶體，以 GB 為單位。	GB (GB)	JobId、JobName、ApplicationId、ApplicationName、WorkerType 和 CapacityAllocationType
WorkerMemoryUsed	工作者在任務執行中所使用的總記憶體，以 GB 為單位。	GB (GB)	JobId、JobName、ApplicationId、ApplicationName、WorkerType 和 CapacityAllocationType
WorkerEphemeralStorageAllocated	任務執行中為工作者配置的暫時性儲存的位元組數。	GB (GB)	JobId、JobName、ApplicationId、ApplicationName、WorkerType 和 CapacityAllocationType
WorkerEphemeralStorageUsed	工作者在任務執行中使用的暫時性儲存的位元組數。	GB (GB)	JobId、JobName、ApplicationId、ApplicationName、WorkerType 和 CapacityAllocationType

指標	Description	單位	維度
WorkerStorageReadBytes	作業執行中工作者從儲存體讀取的位元組數。	位元組	JobId、JobName、ApplicationId、ApplicationName、WorkerType 和 CapacityAllocationType
WorkerStorageWriteBytes	任務執行中從工作者寫入儲存體的位元組數。	位元組	JobId、JobName、ApplicationId、ApplicationName、WorkerType 和 CapacityAllocationType

下列步驟說明如何存取各種類型的指標。

Console

使用主控台存取您的應用程式 UI

1. 透過 [主控台入門](#) 中的指示，導覽至 EMR Studio 上的 EMR Serverless 應用程式。
2. 若要存取執行中任務的引擎特定應用程式 UIs 和日誌：
 - a. 選擇 RUNNING 狀態為 的任務。
 - b. 在應用程式詳細資訊頁面上選取任務，或導覽至任務的任務詳細資訊頁面。
 - c. 在顯示 UI 下拉式功能表下，選擇 Spark UI 或 Hive Tez UI，以導覽至任務類型的應用程式 UI。
 - d. 若要存取 Spark 引擎日誌，請導覽至 Spark UI 中的執行器索引標籤，然後選擇驅動程式的日誌連結。若要存取 Hive 引擎日誌，請在 Hive Tez UI 中選擇適當 DAG 的日誌連結。
3. 若要存取已完成任務的引擎特定應用程式 UIs 和日誌：
 - a. 選擇 SUCCESS 狀態為 的任務。

- b. 在應用程式的應用程式詳細資訊頁面上選取任務，或導覽至任務的任務詳細資訊頁面。
- c. 在顯示 UI 下拉式功能表下，選擇 Spark 歷史記錄伺服器或持久性 Hive Tez UI，以導覽至任務類型的應用程式 UI。
- d. 若要存取 Spark 引擎日誌，請導覽至 Spark UI 中的執行器索引標籤，然後選擇驅動程式的日誌連結。若要存取 Hive 引擎日誌，請在 Hive Tez UI 中選擇適當 DAG 的日誌連結。

AWS CLI

使用 存取您的應用程式 UI AWS CLI

- 若要產生 URL，以使用 存取應用程式 UI 來執行和完成的任務，請呼叫 GetDashboardForJobRun API。

```
aws emr-serverless get-dashboard-for-job-run /  
--application-id <application-id> /  
--job-run-id <job-id>
```

您產生的 URL 有效期為一小時。

使用 Amazon Managed Service for Prometheus 監控 Spark 指標

透過 Amazon EMR 7.1.0 版及更高版本，您可以將 EMR Serverless 與 Amazon Managed Service for Prometheus 整合，以收集 EMR Serverless 任務和應用程式的 Apache Spark 指標。當您使用 AWS 主控台、EMR Serverless API 或 提交任務或建立應用程式時，即可使用此整合 AWS CLI。

先決條件

在您將 Spark 指標交付至 Amazon Managed Service for Prometheus 之前，請先完成下列先決條件。

- [建立 Amazon Managed Service for Prometheus 工作區](#)。此工作區用作擷取端點。記下針對端點 - 遠端寫入 URL 顯示的 URL。建立 EMR Serverless 應用程式時，您需要指定 URL。
- 若要將任務的存取權授予 Amazon Managed Service for Prometheus 以進行監控，請將下列政策新增至您的任務執行角色。

```
{  
  "Sid": "AccessToPrometheus",  
  "Effect": "Allow",  
  "Action": ["aps:RemoteWrite"],
```

```
"Resource": "arn:aws:aps:<AWS_REGION>:<AWS_ACCOUNT_ID>:workspace/<WORKSPACE_ID>"
}
```

設定

使用 AWS 主控台建立與 Amazon Managed Service for Prometheus 整合的應用程式

1. 請參閱 [Amazon EMR Serverless 入門](#) 以建立應用程式。
2. 當您建立應用程式時，請選擇使用自訂設定，然後將資訊指定至您要設定的欄位，以設定您的應用程式。
3. 在應用程式日誌和指標下，選擇將引擎指標交付至 Amazon Managed Service for Prometheus，然後指定遠端寫入 URL。
4. 指定您想要的任何其他組態設定，然後選擇建立和啟動應用程式。

使用 AWS CLI 或 EMR Serverless API

當您執行 AWS CLI 或 `start-job-run` 命令時，您也可以使用 `create-application` 或 EMR Serverless API 將您的 EMR Serverless 應用程式與 Amazon Managed Service for Prometheus 整合。

`create-application`

```
aws emr-serverless create-application \
--release-label emr-7.1.0 \
--type "SPARK" \
--monitoring-configuration '{
  "prometheusMonitoringConfiguration": {
    "remoteWriteUrl": "https://aps-workspaces.<AWS_REGION>.amazonaws.com/
workspaces/<WORKSPACE_ID>/api/v1/remote_write"
  }
}'
```

`start-job-run`

```
aws emr-serverless start-job-run \
--application-id <APPLICATION_ID> \
--execution-role-arn <JOB_EXECUTION_ROLE> \
--job-driver '{
  "sparkSubmit": {
```

```

    "entryPoint": "local:///usr/lib/spark/examples/src/main/python/pi.py",
    "entryPointArguments": ["10000"],
    "sparkSubmitParameters": "--conf spark.dynamicAllocation.maxExecutors=10"
  }
}' \
--configuration-overrides '{
  "monitoringConfiguration": {
    "prometheusMonitoringConfiguration": {
      "remoteWriteUrl": "https://aps-workspaces.<AWS_REGION>.amazonaws.com/
workspaces/<WORKSPACE_ID>/api/v1/remote_write"
    }
  }
}'

```

在您的命令 `prometheusMonitoringConfiguration` 中包含表示 EMR Serverless 必須使用收集 Spark 指標的代理程式來執行 Spark 任務，並將其寫入 Amazon Managed Service for Prometheus 的 `remoteWriteUrl` 端點。然後，您可以使用 Amazon Managed Service for Prometheus 中的 Spark 指標進行視覺化、提醒和分析。

進階組態屬性

EMR Serverless 在 Spark 中使用名為 `PrometheusServlet` 的元件來收集 Spark 指標，並將效能資料轉譯為與 Amazon Managed Service for Prometheus 相容的資料。根據預設，EMR Serverless 會在 Spark 中設定預設值，並在您使用提交任務時剖析驅動程式和執行器指標 `PrometheusMonitoringConfiguration`。

下表說明在提交將指標傳送至 Amazon Managed Service for Prometheus 的 Spark 任務時設定的所有屬性。

Spark 屬性	預設值	Description
<code>spark.metrics.conf</code> <code>.*.sink.prometheusServlet.class</code>	<code>org.apache.spark.metrics.sink.PrometheusServlet</code>	Spark 用來將指標傳送至 Amazon Managed Service for Prometheus 的類別。若要覆寫預設行為，請指定您自己的自訂類別。
<code>spark.metrics.conf</code> <code>.*.source.jvm.class</code>	<code>org.apache.spark.metrics.source.JvmSource</code>	類別 Spark 使用從基礎 Java 虛擬機器收集和傳送關鍵指

Spark 屬性	預設值	Description
		標。若要停止收集 JVM 指標，請將其設定為空字串來停用此屬性，例如 ""。若要覆寫預設行為，請指定您自己的自訂類別。
<code>spark.metrics.conf.driver.sink.prometheusServlet.path</code>	<code>/metrics/prometheus</code>	Amazon Managed Service for Prometheus 用來從驅動程式收集指標的不同 URL。若要覆寫預設行為，請指定您自己的路徑。若要停止收集驅動程式指標，請將其設定為空字串來停用此屬性，例如 ""。
<code>spark.metrics.conf.executor.sink.prometheusServlet.path</code>	<code>/metrics/executor/prometheus</code>	Amazon Managed Service for Prometheus 用來從執行器收集指標的不同 URL。若要覆寫預設行為，請指定您自己的路徑。若要停止收集執行器指標，請將其設定為空字串來停用此屬性，例如 ""。

如需 Spark 指標的詳細資訊，請參閱 [Apache Spark 指標](#)。

考量和限制

使用 Amazon Managed Service for Prometheus 從 EMR Serverless 收集指標時，請考慮下列考量和限制。

- Amazon Managed Service for Prometheus 與 EMR Serverless 搭配使用的支援僅適用於 [AWS 區域 Amazon Managed Service for Prometheus 已全面推出的地方](#)。
- 在 Amazon Managed Service for Prometheus 上執行代理程式以收集 Spark 指標需要更多工作者的資源。如果您選擇較小的工作者大小，例如一個 vCPU 工作者，您的任務執行時間可能會增加。
- Amazon Managed Service for Prometheus 搭配 EMR Serverless 的支援僅適用於 Amazon EMR 7.1.0 版和更新版本。

- Amazon Managed Service for Prometheus 必須部署在您執行 EMR Serverless 以收集指標的相同帳戶中。

EMR Serverless 用量指標

您可以使用 Amazon CloudWatch 用量指標來讓您了解帳戶使用的資源。使用這些指標將 CloudWatch 圖形和儀表板上的服務用量視覺化。

EMR Serverless 用量指標對應至 Service Quotas。您可以設定警示，在您的用量接近服務配額時發出警示。如需詳細資訊，請參閱《[Service Quotas 使用者指南](#)》中的 [Service Quotas](#) 和 [Amazon CloudWatch 警示](#)。Service Quotas

如需 EMR Serverless 服務配額的詳細資訊，請參閱 [端點和配額 EMR Serverless](#)。

EMR Serverless 的服務配額用量指標

EMR Serverless 會在 AWS/Usage 命名空間中發佈下列服務配額用量指標。

指標	Description
ResourceCount	您帳戶上執行的指定資源總數。資源是由與指標相關聯的 維度 所定義。

EMR Serverless 服務配額用量指標的維度

您可以使用下列維度來精簡 EMR Serverless 發佈的用量指標。

維度	Value	Description
Service	EMR Serverless	AWS 服務 包含 資源的 名稱。
Type	資源	EMR Serverless 正在報告的實體類型。
Resource	vCPU	EMR Serverless 正在追蹤的資源類型。

維度	Value	Description
Class	無	EMR Serverless 正在追蹤的資源類別。

使用 自動化 EMR Serverless Amazon EventBridge

您可以使用 Amazon EventBridge 自動化您的 AWS 服務 並自動回應系統事件，例如應用程式可用性問題或資源變更。EventBridge 提供近乎即時的系統事件串流，描述 AWS 資源的變更。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。使用 EventBridge，您可以自動：

- 叫用 AWS Lambda 函數
- 將事件轉送至 Amazon Kinesis Data Streams
- 啟用 AWS Step Functions 狀態機器
- 通知 Amazon SNS 主題或 Amazon SQS 佇列

例如，當您搭配 EMR Serverless 使用 EventBridge 時，您可以在 ETL 任務成功時啟用 AWS Lambda 函數，或在 ETL 任務失敗時通知 Amazon SNS 主題。

EMR Serverless 發出四種類型的事件：

- 應用程式狀態變更事件 – 發出應用程式每個狀態變更的事件。如需應用程式狀態的詳細資訊，請參閱 [應用程式狀態](#)。
- 任務執行狀態變更事件 – 發出任務執行之每個狀態變更的事件。如需的詳細資訊，請參閱 [作業執行狀態](#)。
- 任務執行重試事件 – 從 Amazon EMR Serverless 7.1.0 版和更新版本發出任務執行每次重試的事件。
- 任務資源使用率更新事件 – 以接近 30 分鐘的間隔發出任務執行的資源使用率更新的事件。

EMR Serverless EventBridge 事件範例

EMR Serverless 報告的事件會 `aws.emr-serverless` 指派給 的 `source` 值，如下列範例所示。

應用程式狀態變更事件

下列範例事件顯示CREATING處於 狀態的應用程式。

```
{
  "version": "0",
  "id": "9fd3cf79-1ff1-b633-4dd9-34508dc1e660",
  "detail-type": "EMR Serverless Application State Change",
  "source": "aws.emr-serverless",
  "account": "123456789012",
  "time": "2022-05-31T21:16:31Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "applicationId": "00f1cbsc6anuij25",
    "applicationName": "3965ad00-8fba-4932-a6c8-ded32786fd42",
    "arn": "arn:aws:emr-serverless:us-east-1:111122223333:/
applications/00f1cbsc6anuij25",
    "releaseLabel": "emr-6.6.0",
    "state": "CREATING",
    "type": "HIVE",
    "createdAt": "2022-05-31T21:16:31.547953Z",
    "updatedAt": "2022-05-31T21:16:31.547970Z",
    "autoStopConfig": {
      "enabled": true,
      "idleTimeout": 15
    },
    "autoStartConfig": {
      "enabled": true
    }
  }
}
```

任務執行狀態變更事件

下列範例事件顯示從 SCHEDULED 狀態移至 RUNNING 狀態的任務執行。

```
{
  "version": "0",
  "id": "00df3ec6-5da1-36e6-ab71-20f0de68f8a0",
  "detail-type": "EMR Serverless Job Run State Change",
  "source": "aws.emr-serverless",
  "account": "123456789012",
  "time": "2022-05-31T21:07:42Z",
  "region": "us-east-1",
```

```

    "resources": [],
    "detail": {
      "jobRunId": "00f1cbn5g4bb0c01",
      "applicationId": "00f1982r1uukb925",
      "arn": "arn:aws:emr-serverless:us-east-1:123456789012:/
applications/00f1982r1uukb925/jobruns/00f1cbn5g4bb0c01",
      "releaseLabel": "emr-6.6.0",
      "state": "RUNNING",
      "previousState": "SCHEDULED",
      "createdBy": "arn:aws:sts::123456789012:assumed-role/
TestRole-402dcef3ad14993c15d28263f64381e4cda34775/6622b6233b6d42f59c25dd2637346242",
      "updatedAt": "2022-05-31T21:07:42.299487Z",
      "createdAt": "2022-05-31T21:07:25.325900Z"
    }
  }
}

```

任務執行重試事件

以下是任務執行重試事件的範例。

```

{
  "version": "0",
  "id": "00df3ec6-5da1-36e6-ab71-20f0de68f8a0",
  "detail-type": "EMR Serverless Job Run Retry",
  "source": "aws.emr-serverless",
  "account": "123456789012",
  "time": "2022-05-31T21:07:42Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "jobRunId": "00f1cbn5g4bb0c01",
    "applicationId": "00f1982r1uukb925",
    "arn": "arn:aws:emr-serverless:us-east-1:123456789012:/
applications/00f1982r1uukb925/jobruns/00f1cbn5g4bb0c01",
    "releaseLabel": "emr-6.6.0",
    "createdBy": "arn:aws:sts::123456789012:assumed-role/
TestRole-402dcef3ad14993c15d28263f64381e4cda34775/6622b6233b6d42f59c25dd2637346242",
    "updatedAt": "2022-05-31T21:07:42.299487Z",
    "createdAt": "2022-05-31T21:07:25.325900Z",
    //Attempt Details
    "previousAttempt": 1,
    "previousAttemptState": "FAILED",
    "previousAttemptCreatedAt": "2022-05-31T21:07:25.325900Z",
    "previousAttemptEndedAt": "2022-05-31T21:07:30.325900Z",
  }
}

```

```

    "newAttempt": 2,
    "newAttemptCreatedAt": "2022-05-31T21:07:30.325900Z"
  }
}

```

任務資源使用率更新

下列範例事件顯示任務在執行後移至終端狀態的最終資源使用率更新。

```

{
  "version": "0",
  "id": "00df3ec6-5da1-36e6-ab71-20f0de68f8a0",
  "detail-type": "EMR Serverless Job Resource Utilization Update",
  "source": "aws.emr-serverless",
  "account": "123456789012",
  "time": "2022-05-31T21:07:42Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:emr-serverless:us-east-1:123456789012:/applications/00f1982r1uukb925/jobruns/00f1cbn5g4bb0c01"
  ],
  "detail": {
    "applicationId": "00f1982r1uukb925",
    "jobRunId": "00f1cbn5g4bb0c01",
    "attempt": 1,
    "mode": "BATCH",
    "createdAt": "2022-05-31T21:07:25.325900Z",
    "startedAt": "2022-05-31T21:07:26.123Z",
    "calculatedFrom": "2022-05-31T21:07:42.299487Z",
    "calculatedTo": "2022-05-31T21:07:30.325900Z",
    "resourceUtilizationFinal": true,
    "resourceUtilizationForInterval": {
      "vCPUHour": 0.023,
      "memoryGBHour": 0.114,
      "storageGBHour": 0.228
    },
    "billedResourceUtilizationForInterval": {
      "vCPUHour": 0.067,
      "memoryGBHour": 0.333,
      "storageGBHour": 0
    },
    "totalResourceUtilization": {
      "vCPUHour": 0.023,
      "memoryGBHour": 0.114,

```

```
        "storageGBHour": 0.228
    },
    "totalBilledResourceUtilization": {
        "vCPUHour": 0.067,
        "memoryGBHour": 0.333,
        "storageGBHour": 0
    }
}
}
```

只有在任務已移至執行中狀態時，才會出現 `startedAt` 欄位。

標記資源

使用標籤將您自己的中繼資料指派給每個資源，以協助您管理 EMR Serverless 資源。本節提供標籤函數的概觀，並說明如何建立標籤。

主題

- [什麼是標籤？](#)
- [標記您的 資源](#)
- [標記限制](#)
- [使用 AWS CLI 和 Amazon EMR Serverless API 處理標籤](#)

什麼是標籤？

標籤是您指派給 AWS 資源的標籤。每個標記皆包含由您定義的索引鍵和值。標籤可讓您依用途、擁有者和環境等屬性來分類 AWS 資源。當您有許多相同類型的資源時，請根據指派給該資源的標籤快速識別特定資源。例如，為您的 Amazon EMR Serverless 應用程式定義一組標籤，以協助追蹤每個應用程式的擁有者和堆疊層級。我們建議您為每個資源類型設計一組一致的標籤索引鍵。

標籤不會自動指派給您的資源。將標籤新增至資源後，請修改標籤的值，或隨時從資源中移除標籤。標籤對 Amazon EMR Serverless 沒有任何語意意義，並嚴格解譯為字元字串。若您將與現有標籤具有相同鍵的標籤新增到該資源，則新值會覆寫早前的值。

如果您使用 IAM，您可以控制 AWS 帳戶中哪些使用者具有管理標籤的許可。如需標籤型存取控制政策範例，請參閱 [標籤型存取控制的策略](#)。

標記您的 資源

您可以標記新的或現有的應用程式和任務執行。如果您使用的是 Amazon EMR Serverless API、AWS CLI、或 AWS SDK，您可以使用相關 API 動作上的 tags 參數，將標籤套用至新資源。您也可以使用 TagResource API 動作將標籤套用到現有資源。

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。如果您在建立資源時標記資源，則不需要在建立資源後執行自訂標記指令碼。

下表說明可標記的 Amazon EMR Serverless 資源。

可標記的資源

資源	支援標籤	支援標籤傳播	支援建立時標記 (Amazon EMR Serverless API、AWS CLI 和 AWS SDK)	用於建立的 API (可以在建立過程中新增標籤)
應用程式	是	否。與應用程式相關聯的標籤不會傳播到提交至該應用程式的任務執行。	是	CreateApplication
作業執行	是	否	是	StartJobRun

標記限制

下列基本限制適用於標籤：

- 每個資源最多可有 50 個使用者建立的標籤。
- 針對每一個資源，每個標籤鍵必須是唯一的，且每個標籤鍵只能有一個值。
- 索引鍵的長度上限為 128 個 Unicode 字元 (UTF-8)。
- 值的長度上限則為 256 個 Unicode 字元 (UTF-8)。
- 允許字元為以 UTF-8 表示的字母、數字、空格，以及下列字元：_ . : / = + - @。
- 標籤金鑰不得為空白字串。標籤值可以為空白字串，但不得是 null。
- 標籤鍵與值皆區分大小寫。
- 請勿使用 AWS: 或任何大寫或小寫的組合，例如索引鍵或值的字首。因為僅預留給 AWS 使用。

使用 AWS CLI 和 Amazon EMR Serverless API 處理標籤

使用下列 AWS CLI 命令或 Amazon EMR Serverless API 操作來新增、更新、列出和刪除資源的標籤。

標籤的 CLI 命令和 API 操作

資源	支援標籤	支援標籤傳播
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 標記或取消標記資源 AWS CLI。

標記現有的應用程式

下列命令會標記現有的應用程式。

```
aws emr-serverless tag-resource --resource-arn resource_ARN --tags team=devs
```

取消標記現有應用程式

下列命令會從現有應用程式刪除標籤。

```
aws emr-serverless untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

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-serverless list-tags-for-resource --resource-arn resource_ARN
```

EMR Serverless 教學課程

本節說明使用 EMR Serverless 應用程式時的常見使用案例。這包括各種工具，包括 Hudi 和 Iceberg，用於處理大型資料集，以及使用 Python 和 Python 程式庫提交 Spark 任務。

主題

- [搭配 Amazon EMR Serverless 使用 Java 17](#)
- [搭配 EMR Serverless 使用 Apache Hudi](#)
- [搭配 EMR Serverless 使用 Apache Iceberg](#)
- [搭配 EMR Serverless 使用 Python 程式庫](#)
- [搭配 EMR Serverless 使用不同的 Python 版本](#)
- [搭配 EMR Serverless 使用 Delta Lake OSS](#)
- [從 Airflow 提交 EMR Serverless 任務](#)
- [搭配 EMR Serverless 使用 Hive 使用者定義的函數](#)
- [搭配 EMR Serverless 使用自訂映像](#)
- [在 Amazon EMR Serverless 上使用 Apache Spark 的 Amazon Redshift 整合](#)
- [使用 Amazon EMR Serverless 連線至 DynamoDB](#)

搭配 Amazon EMR Serverless 使用 Java 17

使用 Amazon EMR 6.11.0 版和更新版本，設定 EMR Serverless Spark 任務以使用 Java 虛擬機器 (JVM) 的 Java 17 執行時間。使用下列其中一種方法來使用 Java 17 設定 Spark。

JAVA_HOME

若要覆寫 EMR Serverless 6.11.0 及更高版本的 JVM 設定，請將 JAVA_HOME 設定提供給其 `spark.emr-serverless.driverEnv` 和 `spark.executorEnv` 環境分類。

x86_64

設定必要的屬性，將 Java 17 指定為 Spark 驅動程式和執行器的 JAVA_HOME 組態：

```
--conf spark.emr-serverless.driverEnv.JAVA_HOME=/usr/lib/jvm/java-17-amazon-corretto.x86_64/
```

```
--conf spark.executorEnv.JAVA_HOME=/usr/lib/jvm/java-17-amazon-corretto.x86_64/
```

arm_64

設定必要的屬性，將 Java 17 指定為 Spark 驅動程式和執行器的 JAVA_HOME 組態：

```
--conf spark.emr-serverless.driverEnv.JAVA_HOME=/usr/lib/jvm/java-17-amazon-corretto.aarch64/
--conf spark.executorEnv.JAVA_HOME=/usr/lib/jvm/java-17-amazon-corretto.aarch64/
```

spark-defaults

或者，您可以在 spark-defaults 分類中指定 Java 17，以覆寫 EMR Serverless 6.11.0 和更新版本的 JVM 設定。

x86_64

在 spark-defaults 分類中指定 Java 17：

```
{
  "applicationConfiguration": [
    {
      "classification": "spark-defaults",
      "properties": {
        "spark.emr-serverless.driverEnv.JAVA_HOME" : "/usr/lib/jvm/java-17-amazon-corretto.x86_64/",
        "spark.executorEnv.JAVA_HOME": "/usr/lib/jvm/java-17-amazon-corretto.x86_64/"
      }
    }
  ]
}
```

arm_64

在 spark-defaults 分類中指定 Java 17：

```
{
  "applicationConfiguration": [
    {
      "classification": "spark-defaults",
      "properties": {
```

```

        "spark.emr-serverless.driverEnv.JAVA_HOME" : "/usr/lib/jvm/java-17-
amazon-corretto.aarch64/",
        "spark.executorEnv.JAVA_HOME": "/usr/lib/jvm/java-17-amazon-
corretto.aarch64/"
    }
}
]
}

```

搭配 EMR Serverless 使用 Apache Hudi

本節說明搭配 EMR Serverless 應用程式使用 Apache Hudi。Hudi 是一種資料管理架構，可讓資料處理變得更簡單。

將 Apache Hudi 與 EMR Serverless 應用程式搭配使用

1. 在對應的 Spark 任務執行中設定所需的 Spark 屬性。

```

spark.jars=/usr/lib/hudi/hudi-spark-bundle.jar,/usr/lib/hudi/hudi-utilities-
bundle.jar,/usr/lib/hudi/hudi-aws-bundle.jar
spark.serializer=org.apache.spark.serializer.KryoSerializer

```

2. 若要將 Hudi 資料表同步至設定的目錄，請指定 AWS Glue Data Catalog 做為您的中繼存放區，或設定外部中繼存放區。EMR Serverless 支援 hms 做為 Hudi 工作負載的 Hive 資料表同步模式。EMR Serverless 會將此屬性啟用為預設值。若要進一步了解如何設定中繼存放區，請參閱 [EMR Serverless 的中繼存放區組態](#)。

Important

EMR Serverless 不支援 HIVEQL 或以同步模式選項 JDBC 的形式讓 Hive 資料表處理 Hudi 工作負載。若要進一步了解，請參閱 [同步模式](#)。

當您使用 AWS Glue Data Catalog 做為中繼存放區時，請為您的 Hudi 任務指定下列組態屬性。

```

--conf spark.jars=/usr/lib/hudi/hudi-spark-bundle.jar,
--conf spark.serializer=org.apache.spark.serializer.KryoSerializer,
--conf
spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSG

```

若要進一步了解 Amazon EMR 的 Apache Hudi 版本，請參閱 [Hudi 版本歷史記錄](#)。

搭配 EMR Serverless 使用 Apache Iceberg

本節說明如何將 Apache Iceberg 與 EMR Serverless 應用程式搭配使用。Apache Iceberg 是一種資料表格式，可協助處理資料湖中的大型資料集。

將 Apache Iceberg 與 EMR Serverless 應用程式搭配使用

1. 在對應的 Spark 任務執行中設定所需的 Spark 屬性。

```
spark.jars=/usr/share/aws/iceberg/lib/iceberg-spark3-runtime.jar
```

2. 指定 AWS Glue Data Catalog 做為您的中繼存放區，或設定外部中繼存放區。若要進一步了解如何設定中繼存放區，請參閱 [EMR Serverless 的中繼存放區組態](#)。

設定您要用於 Iceberg 的中繼存放區屬性。例如，如果您想要使用 AWS Glue Data Catalog，請在應用程式組態中設定下列屬性。

```
spark.sql.catalog.dev.warehouse=s3://amzn-s3-demo-bucket/EXAMPLE-PREFIX/  
spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions  
spark.sql.catalog.dev=org.apache.iceberg.spark.SparkCatalog  
spark.sql.catalog.dev.catalog-impl=org.apache.iceberg.aws.glue.GlueCatalog  
spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClient
```

當您使用 AWS Glue Data Catalog 做為中繼存放區時，請為您的 Iceberg 任務指定下列組態屬性。

```
--conf spark.jars=/usr/share/aws/iceberg/lib/iceberg-spark3-runtime.jar,  
--conf  
  spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions,  
--conf spark.sql.catalog.dev=org.apache.iceberg.spark.SparkCatalog,  
--conf spark.sql.catalog.dev.catalog-impl=org.apache.iceberg.aws.glue.GlueCatalog,  
--conf spark.sql.catalog.dev.warehouse=s3://amzn-s3-demo-bucket/EXAMPLE-PREFIX/  
--conf  
  spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClient
```

若要進一步了解 Amazon EMR 的 Apache Iceberg 版本，請參閱 [Iceberg 版本歷史記錄](#)。

搭配 EMR Serverless 使用 Python 程式庫

當您在 Amazon EMR Serverless 應用程式上執行 PySpark 任務時，會將各種 Python 程式庫封裝為相依性。若要這樣做，請使用原生 Python 功能、建置虛擬環境，或直接將 PySpark 任務設定為使用 Python 程式庫。此頁面涵蓋每個方法。

使用原生 Python 功能

當您設定下列組態時，請使用 PySpark 將 Python 檔案 (.py)、壓縮的 Python 套件 (.zip) 和 Egg 檔案 (.egg) 上傳至 Spark 執行器。

```
--conf spark.submit.pyFiles=s3://amzn-s3-demo-bucket/EXAMPLE-PREFIX/<.py|.egg|.zip  
file>
```

如需如何針對 PySpark 任務使用 Python 虛擬環境的詳細資訊，請參閱[使用 PySpark 原生功能](#)。

使用 EMR 筆記本時，您可以執行下列程式碼，在筆記本中提供 Python 相依性：

```
%%configure -f  
{  
  "conf": {  
    "spark.submit.pyFiles": "s3:///amzn-s3-demo-bucket/EXAMPLE-PREFIX/<.py|.egg|.zip  
file>  
  }  
}
```

建置 Python 虛擬環境

若要封裝 PySpark 任務的多個 Python 程式庫，請建立隔離的 Python 虛擬環境。

1. 若要建置 Python 虛擬環境，請使用下列命令。顯示的範例會將套件 `scipy` 和安裝 `matplotlib` 到虛擬環境套件中，並將封存複製到 Amazon S3 位置。

Important

您必須在與 EMR Serverless 中使用的 Python 版本相同的類似 Amazon Linux 2 環境中執行下列命令，也就是 Amazon EMR 6.6.0 版的 Python 3.7.10。您可以在 [EMR Serverless Samples](#) GitHub 儲存庫中找到範例 Dockerfile。

```
# initialize a python virtual environment
python3 -m venv pyspark_venvsource
source pyspark_venvsource/bin/activate

# optionally, ensure pip is up-to-date
pip3 install --upgrade pip

# install the python packages
pip3 install scipy
pip3 install matplotlib

# package the virtual environment into an archive
pip3 install venv-pack
venv-pack -f -o pyspark_venv.tar.gz

# copy the archive to an S3 location
aws s3 cp pyspark_venv.tar.gz s3://amzn-s3-demo-bucket/EXAMPLE-PREFIX/

# optionally, remove the virtual environment directory
rm -fr pyspark_venvsource
```

2. 使用屬性集提交 Spark 任務，以使用 Python 虛擬環境。

```
--conf spark.archives=s3://amzn-s3-demo-bucket/EXAMPLE-PREFIX/
pyspark_venv.tar.gz#environment
--conf spark.emr-serverless.driverEnv.PYSPARK_DRIVER_PYTHON=./environment/bin/
python
--conf spark.emr-serverless.driverEnv.PYSPARK_PYTHON=./environment/bin/python
--conf spark.executorEnv.PYSPARK_PYTHON=./environment/bin/python
```

請注意，如果您不覆寫原始 Python 二進位檔，上一序列設定中的第二個組態將會是 `--conf spark.executorEnv.PYSPARK_PYTHON=python`。

如需如何為 PySpark 任務使用 Python 虛擬環境的詳細資訊，請參閱[使用 Virtualenv](#)。如需如何提交 Spark 任務的更多範例，請參閱[執行 EMR Serverless 任務時使用 Spark 組態](#)。

設定 PySpark 任務以使用 Python 程式庫

使用 Amazon EMR 6.12.0 版及更高版本，您可以直接設定 EMR Serverless PySpark 任務，以使用熱門的資料科學 Python 程式庫，例如 [pandas](#)、[NumPy](#) 和 [PyArrow](#)，而無需任何其他設定。

下列範例示範如何封裝 PySpark 任務的每個 Python 程式庫。

NumPy

NumPy 是適用於科學運算的 Python 程式庫，可為數學、排序、隨機模擬和基本統計資料提供多維陣列和操作。若要使用 NumPy，請執行下列命令：

```
import numpy
```

pandas

pandas 是建置在 NumPy 上的 Python 程式庫。pandas 程式庫為資料科學家提供 [DataFrame](#) 資料結構和資料分析工具。若要使用 pandas，請執行下列命令：

```
import pandas
```

PyArrow

PyArrow 是 Python 程式庫，可管理記憶體內單欄式資料，以提升任務效能。PyArrow 是以 Apache Arrow 跨語言開發規格為基礎，這是以單欄式格式表示和交換資料的標準方法。若要使用 PyArrow，請執行下列命令：

```
import pyarrow
```

搭配 EMR Serverless 使用不同的 Python 版本

除了 [中的使用案例之外](#) [搭配 EMR Serverless 使用 Python 程式庫](#)，您也可以使用 Python 虛擬環境來使用與 Amazon EMR Serverless 應用程式 Amazon EMR 版本中封裝的版本不同的 Python 版本。若要這樣做，請使用您要使用的 Python 版本建置 Python 虛擬環境。

從 Python 虛擬環境提交任務

1. 使用下列範例中的命令建置您的虛擬環境。此範例會將 Python 3.9.9 安裝至虛擬環境套件，並將封存複製到 Amazon S3 位置。

⚠ Important

如果您使用 Amazon EMR 7.0.0 版和更新版本，請在類似於您用於 EMR Serverless 應用程式的 Amazon Linux 2023 環境中執行命令。

如果您使用 6.15.0 版或更低版本，請在類似的 Amazon Linux 2 環境中執行下列命令。

```
# install Python 3.9.9 and activate the venv
yum install -y gcc openssl-devel bzip2-devel libffi-devel tar gzip wget make
wget https://www.python.org/ftp/python/3.9.9/Python-3.9.9.tgz && \
tar xzf Python-3.9.9.tgz && cd Python-3.9.9 && \
./configure --enable-optimizations --enable-shared && \
make altinstall

# create python venv with Python 3.9.9
python3.9 -m venv pyspark_venv_python_3.9.9 --copies
source pyspark_venv_python_3.9.9/bin/activate

# copy system python3 libraries and shared libraries to venv
cp -r /usr/local/lib/python3.9/* ./pyspark_venv_python_3.9.9/lib/python3.9/
cp /usr/local/lib/libpython3.9* ./pyspark_venv_python_3.9.9/lib/

# package venv to archive.
# **Note** that you have to supply --python-prefix option
# to make sure python starts with the path where your
# copied libraries are present.
# Copying the python binary to the "environment" directory.
pip3 install venv-pack
venv-pack -f -o pyspark_venv_python_3.9.9.tar.gz --python-prefix /home/hadoop/
environment

# stage the archive in S3
aws s3 cp pyspark_venv_python_3.9.9.tar.gz s3://<path>

# optionally, remove the virtual environment directory
rm -fr pyspark_venv_python_3.9.9
```

2. 將您的屬性設定為使用 Python 虛擬環境並提交 Spark 任務。

```
# note that the archive suffix "environment" is the same as the directory where you
copied the Python binary.
```

```
--conf spark.archives=s3://amzn-s3-demo-bucket/EXAMPLE-PREFIX/
pyspark_venv_python_3.9.9.tar.gz#environment
--conf spark.emr-serverless.driverEnv.PYSPARK_DRIVER_PYTHON=./environment/bin/
python
--conf spark.emr-serverless.driverEnv.PYSPARK_PYTHON=./environment/bin/python
--conf spark.executorEnv.PYSPARK_PYTHON=./environment/bin/python
--conf spark.emr-serverless.driverEnv.LD_LIBRARY_PATH=./environment/lib
--conf spark.executorEnv.LD_LIBRARY_PATH=./environment/lib
```

如需如何為 PySpark 任務使用 Python 虛擬環境的詳細資訊，請參閱[使用 Virtualenv](#)。如需如何提交 Spark 任務的更多範例，請參閱[執行 EMR Serverless 任務時使用 Spark 組態](#)。

搭配 EMR Serverless 使用 Delta Lake OSS

Amazon EMR 6.9.0 版及更新版本

Note

Amazon EMR 7.0.0 及更高版本使用 Delta Lake 3.0.0，將 `delta-core.jar` 檔案重新命名為 `delta-spark.jar`。如果您使用 Amazon EMR 7.0.0 或更新版本，請務必在組態 `delta-spark.jar` 中指定。

Amazon EMR 6.9.0 及更高版本包含 Delta Lake，因此您不再需要自行封裝 Delta Lake，或為您的 EMR Serverless 任務提供 `--packages` 旗標。

1. 當您提交 EMR Serverless 任務時，請確定您具有下列組態屬性，並在 `sparkSubmitParameters` 欄位中包含下列參數。

```
--conf spark.jars=/usr/share/aws/delta/lib/delta-core.jar,/usr/share/aws/delta/lib/
delta-storage.jar
--conf spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension
--conf
spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog
```

2. 建立本機 `delta_sample.py` 以測試建立和讀取 Delta 資料表。

```
# delta_sample.py
from pyspark.sql import SparkSession
```

```
import uuid

url = "s3://amzn-s3-demo-bucket/delta-lake/output/%s/" % str(uuid.uuid4())
spark = SparkSession.builder.appName("DeltaSample").getOrCreate()

## creates a Delta table and outputs to target S3 bucket
spark.range(5).write.format("delta").save(url)

## reads a Delta table and outputs to target S3 bucket
spark.read.format("delta").load(url).show
```

3. 使用 AWS CLI，將 `delta_sample.py` 檔案上傳至您的 Amazon S3 儲存貯體。然後使用 `start-job-run` 命令將任務提交至現有的 EMR Serverless 應用程式。

```
aws s3 cp delta_sample.py s3://amzn-s3-demo-bucket/code/

aws emr-serverless start-job-run \
  --application-id application-id \
  --execution-role-arn job-role-arn \
  --name emr-delta \
  --job-driver '{
    "sparkSubmit": {
      "entryPoint": "s3://amzn-s3-demo-bucket/code/delta_sample.py",
      "sparkSubmitParameters": "--conf spark.jars=/usr/share/
aws/delta/lib/delta-core.jar,/usr/share/aws/delta/lib/delta-storage.jar --
conf spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension --conf
spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog"
    }
  }'
```

若要搭配 Delta Lake 使用 Python 程式庫，請[封裝程式庫做為相依性](#)，或[使用程式庫做為自訂映像](#)來新增 `delta-core` 程式庫。

或者，您可以使用從 `delta-core` JAR 檔案 `SparkContext.addPyFile` 新增 Python 程式庫：

```
import glob
from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()
spark.sparkContext.addPyFile(glob.glob("/usr/share/aws/delta/lib/delta-core_*.jar")[0])
```

Amazon EMR 6.8.0 版及更低版本

如果您使用的是 Amazon EMR 6.8.0 或更低版本，請遵循下列步驟，將 Delta Lake OSS 與 EMR Serverless 應用程式搭配使用。

1. 若要在 [Amazon EMR Serverless 應用程式中建置與 Spark 版本相容的 Delta Lake](#) 開放原始碼版本，請導覽至 [Delta GitHub](#) 並遵循指示。
2. 將 Delta Lake 程式庫上傳至您 中的 Amazon S3 儲存貯體 AWS 帳戶。
3. 當您在應用程式組態中提交 EMR Serverless 任務時，請包含現在位於儲存貯體中的 Delta Lake JAR 檔案。

```
--conf spark.jars=s3://amzn-s3-demo-bucket/jars/delta-core_2.12-1.1.0.jar
```

4. 為了確保您可以從 Delta 資料表讀取和寫入，請執行範例 PySpark 測試。

```
from pyspark import SparkConf, SparkContext
    from pyspark.sql import HiveContext, SparkSession

    import uuid

    conf = SparkConf()
    sc = SparkContext(conf=conf)
    sqlContext = HiveContext(sc)

    url = "s3://amzn-s3-demo-bucket/delta-lake/output/1.0.1/%s/" %
    str(uuid.uuid4())

    ## creates a Delta table and outputs to target S3 bucket
    session.range(5).write.format("delta").save(url)

    ## reads a Delta table and outputs to target S3 bucket
    session.read.format("delta").load(url).show
```

從 Airflow 提交 EMR Serverless 任務

Apache Airflow 中的 Amazon Provider 提供 EMR Serverless 運算子。如需運算子的詳細資訊，請參閱 Apache Airflow 文件中的 [Amazon EMR Serverless Operators](#)。

您可以使用 `EmrServerlessCreateApplicationOperator` 建立 Spark 或 Hive 應用程式。您也可以使用 `EmrServerlessStartJobOperator` 來啟動一或多個使用新應用程式的任務。

若要搭配 Amazon Managed Workflows for Apache Airflow (MWAA) 搭配 Airflow 2.2.2 使用 運算子，請將以下行新增至您的 `requirements.txt` 檔案，並更新您的 MWAA 環境以使用新的 檔案。

```
apache-airflow-providers-amazon==6.0.0
boto3>=1.23.9
```

請注意，EMR Serverless 支援已新增至 Amazon 供應商的 5.0.0 版。6.0.0 版是與 Airflow 2.2.2 相容的最新版本。您可以在 MWAA 上使用 Airflow 2.4.3 的更新版本。

下列縮寫範例示範如何建立應用程式、執行多個 Spark 任務，然後停止應用程式。[EMR Serverless Samples](#) GitHub 儲存庫提供完整範例。如需 `sparkSubmit` 組態的其他詳細資訊，請參閱 [執行 EMR Serverless 任務時使用 Spark 組態](#)。

```
from datetime import datetime

from airflow import DAG
from airflow.providers.amazon.aws.operators.emr import (
    EmrServerlessCreateApplicationOperator,
    EmrServerlessStartJobOperator,
    EmrServerlessDeleteApplicationOperator,
)

# Replace these with your correct values
JOB_ROLE_ARN = "arn:aws:iam::account-id:role/emr_serverless_default_role"
S3_LOGS_BUCKET = "amzn-s3-demo-bucket"

DEFAULT_MONITORING_CONFIG = {
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {"logUri": f"s3://amzn-s3-demo-bucket/logs/"}
    },
}

with DAG(
    dag_id="example_endtoend_emr_serverless_job",
    schedule_interval=None,
    start_date=datetime(2021, 1, 1),
    tags=["example"],
    catchup=False,
) as dag:
```

```
create_app = EmrServerlessCreateApplicationOperator(
    task_id="create_spark_app",
    job_type="SPARK",
    release_label="emr-6.7.0",
    config={"name": "airflow-test"},
)

application_id = create_app.output

job1 = EmrServerlessStartJobOperator(
    task_id="start_job_1",
    application_id=application_id,
    execution_role_arn=JOB_ROLE_ARN,
    job_driver={
        "sparkSubmit": {
            "entryPoint": "local:///usr/lib/spark/examples/src/main/python/
pi_fail.py",
        }
    },
    configuration_overrides=DEFAULT_MONITORING_CONFIG,
)

job2 = EmrServerlessStartJobOperator(
    task_id="start_job_2",
    application_id=application_id,
    execution_role_arn=JOB_ROLE_ARN,
    job_driver={
        "sparkSubmit": {
            "entryPoint": "local:///usr/lib/spark/examples/src/main/python/pi.py",
            "entryPointArguments": ["1000"]
        }
    },
    configuration_overrides=DEFAULT_MONITORING_CONFIG,
)

delete_app = EmrServerlessDeleteApplicationOperator(
    task_id="delete_app",
    application_id=application_id,
    trigger_rule="all_done",
)

(create_app >> [job1, job2] >> delete_app)
```

搭配 EMR Serverless 使用 Hive 使用者定義的函數

Hive 使用者定義函數 (UDFs) 可讓您建立自訂函數，以處理記錄或記錄群組。在本教學課程中，您將搭配預先存在的 Amazon EMR Serverless 應用程式使用範例 UDF，來執行輸出查詢結果的任務。若要了解如何設定應用程式，請參閱 [Amazon EMR Serverless 入門](#)。

使用 UDF 搭配 EMR Serverless

1. 導覽至 [GitHub](#) 以取得範例 UDF。複製儲存庫並切換到您要使用的 git 分支。將儲存庫 `maven-compiler-plugin pom.xml` 檔案中的更新為具有來源。同時將目標 Java 版本組態更新為 1.8。執行 `mvn package -DskipTests` 以建立包含您範例 UDFs JAR 檔案。
2. 建立 JAR 檔案之後，請使用下列命令將其上傳至 S3 儲存貯體。

```
aws s3 cp brickhouse-0.8.2-JS.jar s3://amzn-s3-demo-bucket/jars/
```

3. 建立範例檔案以使用其中一個範例 UDF 函數。將此查詢儲存為 `udf_example.q` 並上傳至 S3 儲存貯體。

```
add jar s3://amzn-s3-demo-bucket/jars/brickhouse-0.8.2-JS.jar;
CREATE TEMPORARY FUNCTION from_json AS 'brickhouse.udf.json.FromJsonUDF';
select from_json('{"key1":[0,1,2], "key2":[3,4,5,6], "key3":[7,8,9]}', map("",
  array(cast(0 as int))));
select from_json('{"key1":[0,1,2], "key2":[3,4,5,6], "key3":[7,8,9]}', map("",
  array(cast(0 as int))))["key1"][2];
```

4. 提交下列 Hive 任務。

```
aws emr-serverless start-job-run \
  --application-id application-id \
  --execution-role-arn job-role-arn \
  --job-driver '{
    "hive": {
      "query": "s3://amzn-s3-demo-bucket/queries/udf_example.q",
      "parameters": "--hiveconf hive.exec.scratchdir=s3://amzn-s3-demo-bucket/
emr-serverless-hive/scratch --hiveconf hive.metastore.warehouse.dir=s3://'$BUCKET'/
emr-serverless-hive/warehouse"
    }
  }' --configuration-overrides '{
  "applicationConfiguration": [{
    "classification": "hive-site",
    "properties": {
```

```
        "hive.driver.cores": "2",
        "hive.driver.memory": "6G"
    }
}],
"monitoringConfiguration": {
    "s3MonitoringConfiguration": {
        "logUri": "s3://amzn-s3-demo-bucket/logs/"
    }
}
}'
```

5. 使用 `get-job-run` 命令來檢查任務的狀態。等待狀態變更為 SUCCESS。

```
aws emr-serverless get-job-run --application-id application-id --job-run-id job-id
```

6. 使用下列命令下載輸出檔案。

```
aws s3 cp --recursive s3://amzn-s3-demo-bucket/logs/applications/application-id/
jobs/job-id/HIVE_DRIVER/ .
```

`stdout.gz` 檔案如下所示。

```
{"key1": [0, 1, 2], "key2": [3, 4, 5, 6], "key3": [7, 8, 9]}
2
```

搭配 EMR Serverless 使用自訂映像

主題

- [使用自訂 Python 版本](#)
- [使用自訂 Java 版本](#)
- [建置資料科學映像](#)
- [使用 Apache Sedona 處理地理空間資料](#)
- [使用自訂映像的授權資訊](#)

使用自訂 Python 版本

您可以建置自訂映像，以使用不同版本的 Python。例如，若要將 Python 3.10 版用於 Spark 任務，請執行下列命令：

```
FROM public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest

USER root

# install python 3
RUN yum install -y gcc openssl-devel bzip2-devel libffi-devel tar gzip wget make
RUN wget https://www.python.org/ftp/python/3.10.0/Python-3.10.0.tgz && \
tar xzf Python-3.10.0.tgz && cd Python-3.10.0 && \
./configure --enable-optimizations && \
make altinstall

# EMRS runs the image as hadoop
USER hadoop:hadoop
```

提交 Spark 任務之前，請將 屬性設定為使用 Python 虛擬環境，如下所示。

```
--conf spark.emr-serverless.driverEnv.PYSPARK_DRIVER_PYTHON=/usr/local/bin/python3.10
--conf spark.emr-serverless.driverEnv.PYSPARK_PYTHON=/usr/local/bin/python3.10
--conf spark.executorEnv.PYSPARK_PYTHON=/usr/local/bin/python3.10
```

使用自訂 Java 版本

下列範例示範如何建置自訂映像，以針對 Spark 任務使用 Java 11。

```
FROM public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest

USER root

# install JDK 11
RUN amazon-linux-extras install java-openjdk11

# EMRS runs the image as hadoop
USER hadoop:hadoop
```

提交 Spark 任務之前，請將 Spark 屬性設定為使用 Java 11，如下所示。

```
--conf spark.executorEnv.JAVA_HOME=/usr/lib/jvm/java-11-  
openjdk-11.0.16.0.8-1.amzn2.0.1.x86_64  
--conf spark.emr-serverless.driverEnv.JAVA_HOME=/usr/lib/jvm/java-11-  
openjdk-11.0.16.0.8-
```

建置資料科學映像

下列範例示範如何包含常見的資料科學 Python 套件，例如 Pandas 和 NumPy。

```
FROM public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest  
  
USER root  
  
# python packages  
RUN pip3 install boto3 pandas numpy  
RUN pip3 install -U scikit-learn==0.23.2 scipy  
RUN pip3 install sk-dist  
RUN pip3 install xgboost  
  
# EMR Serverless runs the image as hadoop  
USER hadoop:hadoop
```

使用 Apache Sedona 處理地理空間資料

下列範例示範如何建置映像，以包含用於地理空間處理的 Apache Sedona。

```
FROM public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest  
  
USER root  
  
RUN yum install -y wget  
RUN wget https://repo1.maven.org/maven2/org/apache/sedona/sedona-core-3.0_2.12/1.3.0-  
incubating/sedona-core-3.0_2.12-1.3.0-incubating.jar -P /usr/lib/spark/jars/  
RUN pip3 install apache-sedona  
  
# EMRS runs the image as hadoop  
USER hadoop:hadoop
```

使用自訂映像的授權資訊

您可以使用 EMR Serverless 建置自訂映像，以執行特定任務或使用特定版本的軟體套件。自訂映像的修改和分發可能受到規則和授權條款的約束。授權文字會出現在後續的子區段中。

套用至自訂映像的授權

Copyright Amazon.com 及其附屬公司；保留一切權利。本軟體是 [AWS 客戶協議](#) 下 AWS 的內容，未經許可不得散佈。除了 [AWS 智慧財產權授權](#) 中的許可之外，AWS 授權人還授予您以下額外許可：

如果符合下列條件，則允許建立、複製和使用 AWS 內容的衍生項目：

- 您不會修改 AWS 內容本身，任何衍生項目都是您新增內容的嚴格結果。
- 內部重製必須保留上述著作權聲明。
- 本授權條款不允許以來源或二進位形式進行外部分發，無論是否進行修改。

如需使用自訂映像的詳細資訊，請參閱 [搭配 EMR Serverless 使用自訂映像](#)。

在 Amazon EMR Serverless 上使用 Apache Spark 的 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。使用此連接器，在 Amazon EMR Serverless 上使用 Spark 來處理存放在 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 Serverless，[Apache Spark 的 Amazon Redshift 整合](#) 包含為原生整合。

主題

- [使用適用於 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](#)

使用適用於 Apache Spark 的 Amazon Redshift 整合啟動 Spark 應用程式

若要使用與 EMR Serverless 6.9.0 的整合，請將所需的 Spark-Redshift 相依性與 Spark 任務一起傳遞。使用 `--jars` 來包含 Redshift 連接器相關的程式庫。若要存取 `--jars` 選項支援的其他檔案位置，請參閱 Apache Spark 文件的[進階相依性管理](#)一節。

- `spark-redshift.jar`
- `spark-avro.jar`
- `RedshiftJDBC.jar`
- `minimal-json.jar`

Amazon EMR 6.10.0 版及更高版本不需要 `minimal-json.jar` 相依性，並且依預設會向每個叢集自動安裝其他相依性。下列範例示範如何使用 Apache Spark 的 Amazon Redshift 整合啟動 Spark 應用程式。

Amazon EMR 6.10.0 +

使用 EMR Serverless 6.10.0 版和更新版本的 Apache Spark 的 Amazon Redshift 整合，在 Amazon EMR Serverless 上啟動 Spark 任務。

```
spark-submit my_script.py
```

Amazon EMR 6.9.0

若要使用 EMR Serverless 6.9.0 版上 Apache Spark 的 Amazon Redshift 整合在 Amazon EMR Serverless 上啟動 Spark 任務，請使用 `--jars` 選項，如下列範例所示。請注意，與 `--jars` 選項一起列出的路徑是 JAR 檔案的預設路徑。

```
--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
```

```
spark-submit \
  --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 \
  my_script.py
```

使用 Apache Spark 的 Amazon Redshift 整合進行身分驗證

使用 AWS Secrets Manager 擷取登入資料並連線至 Amazon Redshift

您可以安全地向 Amazon Redshift 進行身分驗證，方法是將登入資料存放在 Secrets Manager 中，並讓 Spark 任務呼叫 GetSecretValue API 來擷取登入資料：

```
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
```

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

您可以在 JDBC URL 中指定 Amazon Redshift 資料庫名稱和密碼，以向 Amazon Redshift 叢集驗證 Spark 任務。

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，請使用下列格式在 JDBC URL 中設定使用者名稱和密碼：

```
jdbc:redshift://redshifthost:5439/database?user=username&password=password
```

搭配 Amazon EMR Serverless 任務執行角色使用 IAM 型身分驗證

從 Amazon EMR Serverless 6.9.0 版開始，Amazon Redshift JDBC 驅動程式 2.1 或更新版本會封裝到環境中。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。

反之，請指定 `jdbc:redshift:iam://配置`。這會命令 JDBC 驅動程式使用您的 EMR Serverless 任務執行角色自動擷取登入資料。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](#)。此 URL 的範例如下：

```
jdbc:redshift:iam://examplecluster.abc123xyz789.us-west-2.redshift.amazonaws.com:5439/  
dev
```

當符合提供的條件時，您的任務執行角色需要下列許可：

權限	作業執行角色所需的條件
<code>redshift:GetClusterCredentials</code>	JDBC 驅動器從 Amazon Redshift 獲取憑證時所需的條件
<code>redshift:DescribeCluster</code>	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
<code>redshift-serverless:GetCredentials</code>	JDBC 驅動器從 Amazon Redshift Serverless 獲取憑證時所需的條件
<code>redshift-serverless:GetWorkgroup</code>	如果您使用 Amazon Redshift Serverless，而且要根據工作群組名稱和區域指定 URL，則為必要項目

在不同的 VPC 中連線至 Amazon Redshift

當您在 VPC 下設定佈建的 Amazon Redshift 叢集或 Amazon Redshift Serverless 工作群組時，請為 Amazon EMR Serverless 應用程式設定 VPC 連線，以存取資源。如需如何在 EMR Serverless 應用程式上設定 VPC 連線的詳細資訊，請參閱 [設定 EMR Serverless 應用程式的 VPC 存取以連線至資料](#)。

- 如果您佈建的 Amazon Redshift 叢集或 Amazon Redshift Serverless 工作群組可公開存取，請在建立 EMR Serverless 應用程式時指定一或多個已連接 NAT 閘道的私有子網路。
- 如果您佈建的 Amazon Redshift 叢集或 Amazon Redshift Serverless 工作群組無法公開存取，您必須為 Amazon Redshift 叢集建立 Amazon Redshift 受管 VPC 端點，如 [中所述設定 EMR Serverless 應用程式的 VPC 存取以連線至資料](#)。或者，您可以建立 Amazon Redshift Serverless 工作群組，如 [Amazon Redshift 管理指南中的連線至 Amazon Redshift Serverless](#) 所述。您必須將叢集或子群組與您建立 EMR Serverless 應用程式時指定的私有子網路建立關聯。

Note

如果您使用 IAM 型身分驗證，且 EMR Serverless 應用程式的私有子網路未連接 NAT 閘道，則您還必須在這些子網路上為 Amazon Redshift 或 Amazon Redshift Serverless 建立 VPC 端點。如此一來，JDBC 驅動程式就可以擷取登入資料。

在 Amazon Redshift 中讀取和寫入

下列程式碼範例使用 PySpark，透過資料來源 API 和 SparkSQL，在 Amazon Redshift 資料庫中讀取和寫入範例資料。

Data source API

使用 PySpark，透過資料來源 API，在 Amazon Redshift 資料庫中讀取和寫入範例資料。

```
import boto3
from pyspark.sql import SQLContext

sc = # existing SparkContext
sql_context = SQLContext(sc)

url = "jdbc:redshift:iam://redshifthost:5439/database"
aws_iam_role_arn = "arn:aws:iam::account-id:role/role-name"

df = sql_context.read \
    .format("io.github.spark_redshift_community.spark.redshift") \
    .option("url", url) \
    .option("dbtable", "table-name") \
    .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", "table-name-copy") \
    .option("tempdir", "s3://path/for/temp/data") \
    .option("aws_iam_role", "aws-iam-role-arn") \
    .mode("error") \
    .save()

```

SparkSQL

使用 PySpark，透過 SparkSQL，在 Amazon Redshift 資料庫中讀取和寫入範例資料。

```

import boto3
import json
import sys
import os
from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .enableHiveSupport() \
    .getOrCreate()

url = "jdbc:redshift:iam://redshifthost:5439/database"
aws_iam_role_arn = "arn:aws:iam::account-id:role/role-name"

bucket = "s3://path/for/temp/data"
tableName = "table-name" # Redshift table name

s = f"""CREATE TABLE IF NOT EXISTS {table-name} (country string, data string)
    USING io.github.spark_redshift_community.spark.redshift
    OPTIONS (dbtable '{table-name}', 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(table-name, overwrite=False)
df = spark.sql(f"SELECT * FROM {table-name}")
df.show()
```

使用 Spark 連接器時的考量和限制

- 我們建議您開啟從 Amazon EMR 上的 Spark 到 Amazon Redshift 的 JDBC 連線 SSL。
- 最佳實務是建議您在 中管理 Amazon Redshift 叢集的登入 AWS Secrets Manager 資料。如需範例，請參閱[使用 AWS Secrets Manager 擷取連線至 Amazon Redshift 的登入資料](#)。
- 我們建議您使用 Amazon Redshift 身分驗證參數 `aws_iam_role` 的參數傳遞 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：
 - 我們建議您封鎖對 Amazon Redshift 叢集的公開存取。
 - 我們建議您開啟 [Amazon Redshift 稽核記錄](#)。
 - 我們建議您開啟 [Amazon Redshift 靜態加密](#)。
- 請考慮下列針對 Amazon S3 的建議：
 - 我們建議您封鎖對 Amazon S3 儲存貯體的公開存取。
 - 我們建議您使用 [Amazon S3 伺服器端加密](#) 來加密使用的 Amazon S3 儲存貯體。
 - 我們建議您使用 [Amazon S3 生命週期政策](#) 來定義 Amazon 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。出於安全考慮，我們不支援下列從 Spark 到 Amazon S3 的身分驗證方法：
 - 在 `hadoop-env` 組態分類中設定 AWS 存取金鑰
 - 在 `tempdir` URI 中編碼 AWS 存取金鑰

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 Amazon EMR Serverless 連線至 DynamoDB

在本教學課程中，您將資料子集從[美國地理名稱委員會](#)上傳至 Amazon S3 儲存貯體，然後使用 Amazon EMR Serverless 上的 Hive 或 Spark 將資料複製到 Amazon DynamoDB 資料表進行查詢。

步驟 1：將資料上傳至 Amazon S3 儲存貯體

若要建立 Amazon S3 儲存貯體，請遵循《Amazon Simple Storage Service 主控台使用者指南》中[建立儲存貯體](#)的指示。`amzn-s3-demo-bucket` 將的參考取代為您新建立的儲存貯體名稱。現在，您的 EMR Serverless 應用程式已準備好執行任務。

1. `features.zip` 使用下列命令下載範例資料封存。

```
wget https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/features.zip
```

2. 從封存中解壓縮 `features.txt` 檔案，並存取檔案中的前幾行：

```
unzip features.zip  
head features.txt
```

結果應如下所示。

```
1535908|Big Run|Stream|WV|38.6370428|-80.8595469|794  
875609|Constable Hook|Cape|NJ|40.657881|-74.0990309|7  
1217998|Gooseberry Island|Island|RI|41.4534361|-71.3253284|10  
26603|Boone Moore Spring|Spring|AZ|34.0895692|-111.410065|3681  
1506738|Missouri Flat|Flat|WA|46.7634987|-117.0346113|2605  
1181348|Minnow Run|Stream|PA|40.0820178|-79.3800349|1558  
1288759|Hunting Creek|Stream|TN|36.343969|-83.8029682|1024  
533060|Big Charles Bayou|Bay|LA|29.6046517|-91.9828654|0  
829689|Greenwood Creek|Stream|NE|41.596086|-103.0499296|3671  
541692|Button Willow Island|Island|LA|31.9579389|-93.0648847|98
```

這裡每一行的欄位都指出唯一識別符、名稱、自然特徵類型、狀態、緯度、經度和高度，以英呎為單位。

3. 將您的資料上傳至 Amazon S3

```
aws s3 cp features.txt s3://amzn-s3-demo-bucket/features/
```

步驟 2：建立 Hive 資料表

使用 Apache Spark 或 Hive 建立新的 Hive 資料表，其中包含 Amazon S3 中上傳的資料。

Spark

若要使用 Spark 建立 Hive 資料表，請執行下列命令。

```
import org.apache.spark.sql.SparkSession

val sparkSession = SparkSession.builder().enableHiveSupport().getOrCreate()

sparkSession.sql("CREATE TABLE hive_features \
  (feature_id BIGINT, \
  feature_name STRING, \
  feature_class STRING, \
  state_alpha STRING, \
  prim_lat_dec DOUBLE, \
  prim_long_dec DOUBLE, \
  elev_in_ft BIGINT) \
  ROW FORMAT DELIMITED \
  FIELDS TERMINATED BY '|' \
  LINES TERMINATED BY '\n' \
  LOCATION 's3://amzn-s3-demo-bucket/features';")
```

您現在有一個已填入的 Hive 資料表，其中包含來自 `features.txt` 檔案的資料。若要驗證您的資料是否在資料表中，請執行 Spark SQL 查詢，如下列範例所示。

```
sparkSession.sql(
  "SELECT state_alpha, COUNT(*) FROM hive_features GROUP BY state_alpha;")
```

Hive

若要使用 Hive 建立 Hive 資料表，請執行下列命令。

```
CREATE TABLE hive_features
  (feature_id          BIGINT,
  feature_name        STRING ,
  feature_class       STRING ,
  state_alpha         STRING,
  prim_lat_dec        DOUBLE ,
```

```
prim_long_dec      DOUBLE ,
elev_in_ft         BIGINT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
LINES TERMINATED BY '\n'
LOCATION 's3://amzn-s3-demo-bucket/features';
```

您現在有一個 Hive 資料表，其中包含來自 `features.txt` 檔案的資料。若要驗證您的資料是否在資料表中，請執行 HiveQL 查詢，如下列範例所示。

```
SELECT state_alpha, COUNT(*) FROM hive_features GROUP BY state_alpha;
```

步驟 3：將資料複製到 DynamoDB

使用 Spark 或 Hive 將資料複製到新的 DynamoDB 資料表。

Spark

若要將您在上一個步驟中建立的 Hive 資料表中的資料複製到 DynamoDB，請遵循[將資料複製到 DynamoDB](#) 中的步驟 1-3。這會建立新的 DynamoDB 資料表，稱為 `Features`。然後，您可以直接從文字檔案讀取資料，並將其複製到 DynamoDB 資料表，如下列範例所示。

```
import com.amazonaws.services.dynamodbv2.model.AttributeValue
import org.apache.hadoop.dynamodb.DynamoDBItemWritable
import org.apache.hadoop.dynamodb.read.DynamoDBInputFormat
import org.apache.hadoop.io.Text
import org.apache.hadoop.mapred.JobConf
import org.apache.spark.SparkContext

import scala.collection.JavaConverters._

object EmrServerlessDynamoDbTest {

  def main(args: Array[String]): Unit = {

    jobConf.set("dynamodb.input.tableName", "Features")
    jobConf.set("dynamodb.output.tableName", "Features")
    jobConf.set("dynamodb.region", "region")

    jobConf.set("mapred.output.format.class",
"org.apache.hadoop.dynamodb.write.DynamoDBOutputFormat")
```

```
jobConf.set("mapred.input.format.class",
"org.apache.hadoop.dynamodb.read.DynamoDBInputFormat")

val rdd = sc.textFile("s3://amzn-s3-demo-bucket/ddb-connector/")
  .map(row => {
    val line = row.split("\\|")
    val item = new DynamoDBItemWritable()

    val elevInFt = if (line.length > 6) {
      new AttributeValue().withN(line(6))
    } else {
      new AttributeValue().withNULL(true)
    }

    item.setItem(Map(
      "feature_id" -> new AttributeValue().withN(line(0)),
      "feature_name" -> new AttributeValue(line(1)),
      "feature_class" -> new AttributeValue(line(2)),
      "state_alpha" -> new AttributeValue(line(3)),
      "prim_lat_dec" -> new AttributeValue().withN(line(4)),
      "prim_long_dec" -> new AttributeValue().withN(line(5)),
      "elev_in_ft" -> elevInFt)
      .asJava)
      (new Text(""), item)
    })
  rdd.saveAsHadoopDataset(jobConf)
}
```

Hive

若要將您在上一個步驟中建立的 Hive 資料表中的資料複製到 DynamoDB，請遵循[將資料複製到 DynamoDB](#) 中的指示。

步驟 4：從 DynamoDB 查詢資料

使用 Spark 或 Hive 查詢 DynamoDB 資料表。

Spark

若要從您在上一個步驟中建立的 DynamoDB 資料表查詢資料，請使用 Spark SQL 或 Spark MapReduce API。

Example– 使用 Spark SQL 查詢 DynamoDB 資料表

下列 Spark SQL 查詢會依字母順序傳回所有功能類型的清單。

```
val dataframe = sparkSession.sql("SELECT DISTINCT feature_class \
FROM ddb_features \
ORDER BY feature_class;")
```

下列 Spark SQL 查詢會傳回以字母 M 開頭的所有湖的清單。

```
val dataframe = sparkSession.sql("SELECT feature_name, state_alpha \
FROM ddb_features \
WHERE feature_class = 'Lake' \
AND feature_name LIKE 'M%' \
ORDER BY feature_name;")
```

下列 Spark SQL 查詢會傳回所有狀態的清單，其中包含至少三個高於一英里的功能。

```
val dataframe = sparkSession.dql("SELECT state_alpha, feature_class, COUNT(*) \
FROM ddb_features \
WHERE elev_in_ft > 5280 \
GROUP BY state_alpha, feature_class \
HAVING COUNT(*) >= 3 \
ORDER BY state_alpha, feature_class;")
```

Example– 使用 Spark MapReduce API 查詢 DynamoDB 資料表

下列 MapReduce 查詢會依字母順序傳回所有功能類型的清單。

```
val df = sc.hadoopRDD(jobConf, classOf[DynamoDBInputFormat], classOf[Text],
classOf[DynamoDBItemWritable])
  .map(pair => (pair._1, pair._2.getItem))
  .map(pair => pair._2.get("feature_class").getS)
  .distinct()
  .sortBy(value => value)
  .toDF("feature_class")
```

下列 MapReduce 查詢會傳回以字母 M 開頭的所有湖的清單。

```
val df = sc.hadoopRDD(jobConf, classOf[DynamoDBInputFormat], classOf[Text],
classOf[DynamoDBItemWritable])
```

```

.map(pair => (pair._1, pair._2.getItem))
.filter(pair => "Lake".equals(pair._2.get("feature_class").getS))
.filter(pair => pair._2.get("feature_name").getS.startsWith("M"))
.map(pair => (pair._2.get("feature_name").getS,
pair._2.get("state_alpha").getS))
.sortBy(_._1)
.toDF("feature_name", "state_alpha")

```

下列 MapReduce 查詢會傳回所有狀態的清單，其中包含至少三個高於一英里的功能。

```

val df = sc.hadoopRDD(jobConf, classOf[DynamoDBInputFormat], classOf[Text],
classOf[DynamoDBItemWritable])
.map(pair => pair._2.getItem)
.filter(pair => pair.get("elev_in_ft").getN != null)
.filter(pair => Integer.parseInt(pair.get("elev_in_ft").getN) > 5280)
.groupBy(pair => (pair.get("state_alpha").getS, pair.get("feature_class").getS))
.filter(pair => pair._2.size >= 3)
.map(pair => (pair._1._1, pair._1._2, pair._2.size))
.sortBy(pair => (pair._1, pair._2))
.toDF("state_alpha", "feature_class", "count")

```

Hive

若要從您在上一個步驟中建立的 DynamoDB 資料表查詢資料，請遵循 [DynamoDB 資料表中查詢資料](#) 中的指示。

設定跨帳戶存取權

若要設定 EMR Serverless 的跨帳戶存取，請完成下列步驟。在此範例中，AccountA 是您建立 Amazon EMR Serverless 應用程式的帳戶，而 AccountB 是 Amazon DynamoDB 所在的帳戶。

1. 在 AccountA 中建立 DynamoDB 資料表 AccountB。如需詳細資訊，請參閱 [步驟 1：建立資料表](#)。
2. 在 Cross-Account-Role-B 中建立可存取 DynamoDB 資料表的 AccountB IAM 角色。
 - a. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
 - b. 選擇角色，然後建立名為 Cross-Account-Role-B 的新角色。如需如何建立 IAM 角色的詳細資訊，請參閱《使用者指南》中的 [建立 IAM 角色](#)。
 - c. 建立 IAM 政策，授予存取跨帳戶 DynamoDB 資料表的許可。然後，將 IAM 政策附接至 Cross-Account-Role-B。

以下是授予 DynamoDB 資料表 存取權的政策CrossAccountTable。

- d. 編輯 Cross-Account-Role-B 角色的信任關係。

若要設定角色的信任關係，請在 IAM 主控台中為您**在**步驟 2 : Cross-Account-Role-B 中建立的角色選擇信任關係索引標籤。

選取編輯信任關係，然後新增下列政策文件。本文件允許 Job-Execution-Role-A中的 AccountA擔任此Cross-Account-Role-B角色。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSTSAssumerole",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/Job-Execution-Role-A"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- e. 授予 Job-Execution-Role-A AccountA 以擔任的- STS Assume role許可Cross-Account-Role-B。

在的 IAM 主控台中 AWS 帳戶 AccountA，選取 Job-Execution-Role-A。將以下政策陳述式新增至 Job-Execution-Role-A 以允許 Cross-Account-Role-B 角色的 AssumeRole 動作。

JSON

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

```

    "Action": [
      "sts:AssumeRole"
    ],
    "Resource": [
      "arn:aws:iam::123456789012:role/Cross-Account-Role-B"
    ],
    "Sid": "AllowSTSAssumerole"
  }
]
}

```

- f. 將 `dynamodb.customAWSCredentialsProvider` 屬性的值設定為核心網站分類 `com.amazonaws.emr.AssumeRoleAWSCredentialsProvider` 中的。設定 ARN 值 `ASSUME_ROLE_CREDENTIALS_ROLE_ARN` 為的環境變數 `Cross-Account-Role-B`。

3. 使用執行 Spark 或 Hive 任務 `Job-Execution-Role-A`。

考量事項

當您搭配 Apache Spark 或 Apache Hive 使用 DynamoDB 連接器時，請注意這些行為和限制。

搭配 Apache Spark 使用 DynamoDB 連接器時的考量事項

- Spark SQL 不支援使用儲存處理器選項建立 Hive 資料表。如需詳細資訊，請參閱 Apache Spark 文件中的 [指定 Hive 資料表的儲存格式](#)。
- Spark SQL 不支援使用儲存處理常式 `STORED BY` 的操作。如果您想要透過外部 Hive 資料表與 DynamoDB 資料表互動，請先使用 Hive 建立資料表。
- 若要將查詢轉譯為 DynamoDB 查詢，DynamoDB 連接器會使用述詞下推。述詞下推依對應至 DynamoDB 資料表分割區索引鍵的資料欄篩選資料。述詞下推只有在您將連接器與 Spark SQL 搭配使用時才會運作，而不是與 MapReduce API 搭配使用時。

搭配 Apache Hive 使用 DynamoDB 連接器時的考量事項

調校映射器的最大數量

- 如果您使用 `SELECT` 查詢從映射至 DynamoDB 的外部 Hive 資料表讀取資料，EMR Serverless 上的映射任務數量會計算為針對 DynamoDB 資料表設定的讀取總輸送量，除以每個映射任務的輸送量。每個映射任務的預設輸送量為 100。

- Hive 任務可以使用超過每個 EMR Serverless 應用程式設定的容器數量上限的映射任務數量，取決於為 DynamoDB 設定的讀取輸送量。此外，長時間執行的 Hive 查詢可能會耗用 DynamoDB 資料表的所有佈建讀取容量。這會對其他使用者造成負面影響。
- 您可以使用 `dynamodb.max.map.tasks` 屬性來設定映射任務的上限。您也可以使用此屬性，根據任務容器大小來調整每個映射任務讀取的資料量。
- 您可以在 Hive 查詢層級或在 `start-job-run` 命令的 `hive-site` 分類中設定 `dynamodb.max.map.tasks` 屬性。此數值必須等於或大於 1。當 Hive 處理您的查詢時，產生的 Hive 任務使用的值不超過從 DynamoDB 資料表讀取 `dynamodb.max.map.tasks` 時的值。

調整每個任務的寫入輸送量

- EMR Serverless 上每個任務的寫入輸送量計算方式為為 DynamoDB 資料表設定的總寫入輸送量，除以 `mapreduce.job.maps` 屬性的值。對於 Hive，此屬性的預設值為 2。因此，Hive 任務最後階段的前兩個任務可以使用所有寫入輸送量。這會導致調節相同任務或其他任務中其他任務的寫入。
- 若要避免寫入限流，請根據最後一個階段的任務數量或每個任務要配置的寫入輸送量，設定 `mapreduce.job.maps` 屬性的值。在 EMR Serverless 上的 `start-job-run` 命令 `mapred-site` 分類中設定此屬性。

安全

的雲端安全性 AWS 是最高優先順序。身為 AWS 客戶，您可以受益於資料中心和網路架構，該架構專為滿足最安全敏感組織的需求而建置。

安全性是 AWS 與您之間共同責任。[共同責任模型](#) 將此描述為雲端的安全和雲端內的安全：

- 雲端的安全性 – AWS 負責保護在 Cloud AWS 中執行 AWS 服務的基礎設施。AWS 也提供您安全使用的服務。在 [AWS 合規計畫](#) 中，第三方稽核員會定期測試並驗證我們的安全功效。若要了解適用於 Amazon EMR Serverless 的合規計畫，請參閱 [AWS 合規計畫範圍內的服務](#)。
- 雲端的安全性 – 您的責任取決於您使用 AWS 的服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件可協助您了解如何在使用 Amazon EMR Serverless 時套用共同責任模型。這些主題說明如何設定 Amazon EMR Serverless 並使用其他服務 AWS 來滿足您的安全和合規目標。

主題

- [Amazon EMR Serverless 的安全最佳實務](#)
- [資料保護](#)
- [Amazon EMR Serverless 中的 Identity and Access Management \(IAM\)](#)
- [受信任的身分傳播](#)
- [搭配 EMR Serverless 使用 Lake Formation](#)
- [工作者間加密](#)
- [使用 KMS CMK 進行磁碟加密](#)
- [使用 EMR Serverless 進行資料保護的 Secrets Manager](#)
- [搭配 EMR Serverless 使用 Amazon S3 Access Grants](#)
- [使用 記錄 Amazon EMR Serverless API 呼叫 AWS CloudTrail](#)
- [Amazon EMR Serverless 的合規驗證](#)
- [Amazon EMR Serverless 中的彈性](#)
- [Amazon EMR Serverless 中的基礎設施安全性](#)
- [Amazon EMR Serverless 中的組態和漏洞分析](#)

Amazon EMR Serverless 的安全最佳實務

Amazon EMR Serverless 提供多種安全功能，供您在開發和實作自己的安全政策時加以考量。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。

套用最低權限準則

EMR Serverless 為使用 IAM 角色的應用程式提供精細存取政策，例如執行角色。我們建議僅授予執行角色任務所需的最低權限集，例如涵蓋您的應用程式和對日誌目的地的存取。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。

隔離不受信任的應用程式碼

EMR Serverless 會在屬於不同 EMR Serverless 應用程式的任務之間建立完整的網路隔離。在需要任務層級隔離的情況下，請考慮將任務隔離到不同的 EMR Serverless 應用程式。

角色型存取控制 (RBAC) 許可

管理員應嚴格控制 EMR Serverless 應用程式的角色型存取控制 (RBAC) 許可。

資料保護

AWS [共同責任模型](#)適用於 Amazon EMR Serverless 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。您負責維護在此基礎設施上託管內容的控制權。此內容包含您使用之 AWS 服務的安全組態和管理任務。如需資料隱私權的詳細資訊，請參閱[資料隱私權常見問答集](#)。如需歐洲資料保護的相關資訊，請參閱 AWS 安全部落格上的[AWS 共同責任模型和 GDPR](#) 部落格文章。

基於資料保護目的，我們建議您保護 AWS 帳戶登入資料，並使用 AWS Identity and Access Management (IAM) 設定個別帳戶。如此一來，每個使用者都只會獲得授予完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源通訊。我們建議使用 TLS 1.2 或更新版本。
- 使用設定 API 和使用者活動記錄 AWS CloudTrail。
- 使用 AWS 加密解決方案，以及服務中的所有 AWS 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 Serverless 加密選項來加密靜態和傳輸中的資料。
- 如果您在 AWS 透過命令列界面或 API 存取 時需要 FIPS 140-2 驗證的密碼編譯模組 , 請使用 FIPS 端點。如需可用 FIPS 端點的詳細資訊 , 請參閱 [聯邦資訊處理標準 \(FIPS\) 140-2](#)。

我們強烈建議您絕對不要將敏感的識別資訊 , 例如客戶的帳戶號碼 , 放入自由格式欄位 , 例如名稱欄位。這包括當您使用 Amazon EMR Serverless 或使用主控台 AWS CLI、API 或 AWS SDKs 的其他 AWS 服務時。您在 Amazon EMR Serverless 或其他 服務中輸入的任何資料都可能被選入診斷日誌中。當您提供外部伺服器的 URL 時 , 請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

靜態加密

資料加密有助於防止未經授權的使用者讀取叢集上的資料和相關的資料儲存體系統。這包括儲存到持久性媒體的資料 (稱為靜態資料) , 以及透過網路傳送時可能會被攔截的資料 (稱為傳輸中資料)。

資料加密需要金鑰和憑證。您可以從數個選項中選擇 , 包括 管理的金鑰 AWS Key Management Service、Amazon S3 管理的金鑰 , 以及您提供的自訂提供者的金鑰和憑證。使用 AWS KMS 做為金鑰提供者時 , 會收取儲存和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

每個 EMR Serverless 應用程式都使用特定的發行版本 , 其中包括 EMRFS (EMR 檔案系統)。Amazon S3 加密適用於讀取和寫入至 Amazon S3 的 EMR 檔案系統 (EMRFS) 物件。當您啟用靜態加密時 , 您可以將 Amazon S3 伺服器端加密 (SSE) 或用戶端加密 (CSE) 指定為預設加密模式。或者 , 使用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。如果您使用 Amazon S3 CSE 搭配客戶受管金鑰 , 則用於在 EMR Serverless 應用程式中執行任務的執行角色必須能夠存取金鑰。如需 Amazon S3 加密的深入資訊 , 請參閱《Amazon Simple Storage Service 開發人員指南》中的 [使用加密保護資料](#)。

Note

當您使用 時 AWS KMS , 會收取儲存和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

所有 Amazon S3 儲存貯體都已預設設定加密，且所有上傳至 S3 儲存貯體的新物件都會在靜態時自動加密，Amazon S3 會在將資料寫入磁碟時加密物件層級的資料，並在存取資料時解密資料。如需 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 Serverless 中指定 SSE 時，您可以在兩個不同的金鑰管理系統之間進行選擇：

- SSE-S3 – Amazon S3 為您管理密鑰。EMR Serverless 不需要額外的設定。
- SSE-KMS - 您可以使用 AWS KMS key 來設定適用於 EMR Serverless 的政策。EMR Serverless 不需要額外的設定。

Tip

若要在使用 SSE-KMS AWS KMS 時降低成本，請考慮在 Amazon S3 儲存貯體上啟用 Amazon S3 儲存貯體金鑰。Amazon S3 儲存貯體金鑰使用短期儲存貯體層級金鑰，可將 AWS KMS API 呼叫減少高達 99%。啟用 Amazon S3 儲存貯體金鑰之前，請檢閱您的 IAM 和 AWS KMS 金鑰政策 – 加密內容會從 Amazon S3 物件 ARN 變更為儲存貯體 ARN，這可能會影響使用物件 ARN 進行存取控制的政策。如需詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的[使用 Simple Storage Service \(Amazon S3\) 儲存貯體金鑰降低 SSE-KMS 的成本](#)。

若要對您寫入 Amazon S3 的資料使用 AWS KMS 加密，當您使用 StartJobRun API 時有兩個選項。您可以為寫入 Amazon S3 的所有項目啟用加密，或為寫入特定儲存貯體的資料啟用加密。如需 StartJobRun API 的詳細資訊，請參閱[EMR Serverless API 參考](#)。

若要為您寫入 Amazon S3 的所有資料開啟 AWS KMS 加密，請在呼叫 StartJobRun API 時使用下列命令。

```
--conf spark.hadoop.fs.s3.enableServerSideEncryption=true  
--conf spark.hadoop.fs.s3.serverSideEncryption.kms.keyId=<kms_id>
```

若要為寫入特定儲存貯體的資料開啟 AWS KMS 加密，請在呼叫 StartJobRun API 時使用下列命令。

```
--conf spark.hadoop.fs.s3.bucket.<amzn-s3-demo-bucket1>.enableServerSideEncryption=true
```

```
--conf spark.hadoop.fs.s3.bucket.<amzn-s3-demo-  
bucket1>.serverSideEncryption.kms.keyId=<kms-id>
```

使用客戶提供的金鑰 (SSE-C) 的 SSE 無法與 EMR Serverless 搭配使用。

Amazon S3 用戶端加密

透過 Amazon S3 用戶端加密，Amazon S3 加密和解密會在每個 Amazon EMR 版本上可用的 EMRFS 用戶端中進行。物件在上傳至 Amazon S3 之前會先加密，並在下載後解密。您指定的提供者會提供用戶端使用的加密金鑰。用戶端可以使用 AWS KMS (CSE-KMS) 提供的金鑰或提供用戶端根金鑰 (CSE-C) 的自訂 Java 類別。CSE-KMS 和 CSE-C 之間的加密細節略有不同，具體取決於指定的提供者和要解密或加密之物件的中繼資料。如果您使用 Amazon S3 CSE 搭配客戶受管金鑰，則用於在 EMR Serverless 應用程式中執行任務的執行角色必須能夠存取金鑰。可能需要支付額外的 KMS 費用。如需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](#)。

本機磁碟加密

存放在暫時性儲存的資料會使用業界標準的 AES-256 密碼編譯演算法，以服務擁有的金鑰加密。

金鑰管理

可以將 KMS 設定為自動輪換 KMS 金鑰。這將每年輪換一次金鑰，同時無限期儲存舊金鑰，以便您的資料仍然可以解密。如需詳細資訊，請參閱[輪換客戶受管金鑰](#)。

傳輸中加密

Amazon EMR Serverless 提供下列應用程式特定的加密功能：

- Spark
 - 根據預設，Spark 驅動程式和執行器之間的通訊會經過身分驗證且為內部。驅動程式和執行器之間的 RPC 通訊已加密。
- Hive
 - AWS Glue 中繼存放區與 EMR Serverless 應用程式之間的通訊會透過 TLS 進行。

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。

Amazon EMR Serverless 中的 Identity and Access Management (IAM)

AWS Identity and Access Management (IAM) 是 AWS 服務，可協助管理員安全地控制對 AWS 資源的存取。IAM 管理員可控制誰可以進行身分驗證（登入）和授權（具有許可），以使用 Amazon EMR Serverless 資源。IAM 是您可以免費使用 AWS 服務的。

主題

- [目標對象](#)
- [使用身分驗證](#)
- [使用政策管理存取權](#)
- [EMR Serverless 如何與 IAM 搭配使用](#)
- [使用 EMR Serverless 的服務連結角色](#)
- [Amazon EMR Serverless 的任務執行期角色](#)
- [EMR Serverless 的使用者存取政策範例](#)
- [標籤型存取控制的策略](#)
- [EMR Serverless 的身分型政策範例](#)
- [AWS 受管政策的 Amazon EMR Serverless 更新](#)
- [對 Amazon EMR Serverless 身分和存取進行故障診斷](#)

目標對象

使用方式 AWS Identity and Access Management (IAM) 會根據您的角色而有所不同：

- 服務使用者 — 若無法存取某些功能，請向管理員申請所需許可 (請參閱 [對 Amazon EMR Serverless 身分和存取進行故障診斷](#))
- 服務管理員 — 負責設定使用者存取權並提交相關許可請求 (請參閱 [Amazon EMR Serverless 中的 Identity and Access Management \(IAM\)](#))
- IAM 管理員 — 撰寫政策以管理存取控制 (請參閱 [EMR Serverless 的身分型政策範例](#))

使用身分驗證

身分驗證是您 AWS 使用身分憑證登入的方式。您必須以 AWS 帳戶根使用者、IAM 使用者或擔任 IAM 角色身分進行身分驗證。

您可以使用身分來源的登入資料，例如 AWS IAM Identity Center (IAM Identity Center)、單一登入身分驗證或 Google/Facebook 登入資料，以聯合身分的形式登入。如需有關登入的詳細資訊，請參閱《AWS 登入 使用者指南》中的[如何登入您的 AWS 帳戶](#)。

對於程式設計存取，AWS 提供 SDK 和 CLI 以密碼編譯方式簽署請求。如需詳細資訊，請參閱《IAM 使用者指南》中的[API 請求的 AWS 第 4 版簽署程序](#)。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個名為 AWS 帳戶 theroot 使用者的登入身分開始，該身分可完整存取所有 AWS 服務和資源。強烈建議不要使用根使用者來執行日常任務。有關需要根使用者憑證的任務，請參閱《IAM 使用者指南》中的[需要根使用者憑證的任務](#)。

聯合身分

最佳實務是要求人類使用者使用聯合身分提供者，以 AWS 服務使用臨時憑證存取。

聯合身分是您企業目錄、Web 身分提供者的使用者，或使用身分來源的 AWS 服務憑證存取 Directory Service。聯合身分會擔任角色，而該角色會提供臨時憑證。

若需集中化管理存取權限，建議使用 AWS IAM Identity Center。如需詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的[什麼是 IAM Identity Center?](#)。

IAM 使用者和群組

IAM 使用者https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html是一種身分具備單人或應用程式的特定許可權。建議以臨時憑證取代具備長期憑證的 IAM 使用者。如需詳細資訊，請參閱《IAM 使用者指南》中的[要求人類使用者使用聯合身分提供者來 AWS 使用臨時憑證存取](#)。

[IAM 群組](#)會指定 IAM 使用者集合，使管理大量使用者的許可權更加輕鬆。如需詳細資訊，請參閱《IAM 使用者指南》中的[IAM 使用者的使用案例](#)。

IAM 角色

IAM 角色https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html的身分具有特定許可權，其可以提供臨時憑證。您可以透過[從使用者切換到 IAM 角色 \(主控台\)](#)或呼叫 AWS CLI 或 AWS API 操作來擔任角色。如需詳細資訊，請參閱《IAM 使用者指南》中的[擔任角色的方法](#)。

IAM 角色適用於聯合身分使用者存取、臨時 IAM 使用者許可、跨帳戶存取權與跨服務存取，以及在 Amazon EC2 執行的應用程式。如需詳細資訊，請參閱《IAM 使用者指南》中的[IAM 中的快帳戶資源存取](#)。

使用政策管理存取權

您可以透過建立政策並將其連接到身分或資源 AWS 來控制 AWS 中的存取。政策定義與身分或資源相關聯的許可。當委託人提出請求時 AWS，會評估這些政策。大多數政策會以 JSON 文件 AWS 形式存放在中。如需進一步了解 JSON 政策文件，請參閱《IAM 使用者指南》中的 [JSON 政策概觀](#)。

管理員會使用政策，透過定義哪些主體可在哪些條件下對哪些資源執行動作，以指定可存取的範圍。

預設情況下，使用者和角色沒有許可。IAM 管理員會建立 IAM 政策並將其新增至角色，供使用者後續擔任。IAM 政策定義動作的許可，無論採用何種方式執行。

身分型政策

身分型政策是附加至身分 (使用者、使用者群組或角色) 的 JSON 許可政策文件。這類政策控制身分可對哪些資源執行哪些動作，以及適用的條件。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的 [透過客戶管理政策定義自訂 IAM 許可](#)。

身分型政策可分為內嵌政策 (直接內嵌於單一身分) 與受管政策 (可附加至多個身分的獨立政策)。如需了解如何在受管政策及內嵌政策之間做選擇，請參閱《IAM 使用者指南》中的 [在受管政策與內嵌政策之間選擇](#)。

資源型政策

資源型政策是附加到資源的 JSON 政策文件。範例包括 IAM 角色信任政策與 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。您必須在資源型政策中 [指定主體](#)。

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

其他政策類型

AWS 支援其他政策類型，可設定更多常見政策類型授予的最大許可：

- 許可界限 — 設定身分型政策可授與 IAM 實體的最大許可。如需詳細資訊，請參閱《IAM 使用者指南》中的 [IAM 實體許可界限](#)。
- 服務控制政策 (SCP) — 為 AWS Organizations 中的組織或組織單位指定最大許可。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的 [服務控制政策](#)。
- 資源控制政策 (RCP) — 設定您帳戶中資源可用許可的上限。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的 [資源控制政策 \(RCP\)](#)。

- 工作階段政策 — 在以程式設計方式為角色或聯合身分使用者建立臨時工作階段時，以參數形式傳遞的進階政策。如需詳細資訊，請參閱《IAM 使用者指南》中的[工作階段政策](#)。

多種政策類型

當請求套用多種類型的政策時，產生的許可會更複雜而無法理解。若要了解如何 AWS 決定是否在涉及多個政策類型時允許請求，請參閱《IAM 使用者指南》中的[政策評估邏輯](#)。

EMR Serverless 如何與 IAM 搭配使用

在您使用 IAM 管理對 Amazon EMR Serverless 的存取之前，請先了解哪些 IAM 功能可與 Amazon EMR Serverless 搭配使用。

搭配 EMR Serverless 使用的 IAM 功能

IAM 功能	Amazon EMR Serverless 支援
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	否
ACL	否
ABAC (政策中的標籤)	是
臨時憑證	是
主體許可	是
服務角色	否
服務連結角色	是

若要全面了解 EMR Serverless 和其他 AWS 服務如何與大多數 IAM 功能搭配使用，請參閱《[AWS IAM 使用者指南](#)》中的[與 IAM 搭配使用的服務](#)。

EMR Serverless 的身分型政策

支援身分型政策：是

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的[透過客戶管理政策定義自訂 IAM 許可](#)。

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。如要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的[IAM JSON 政策元素參考](#)。

EMR Serverless 的身分型政策範例

若要存取 Amazon EMR Serverless 身分型政策的範例，請參閱[EMR Serverless 的身分型政策範例](#)。

EMR Serverless 中的資源型政策

支援資源型政策：否

資源型政策是附加到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中[指定主體](#)。委託人可以包含帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

如需啟用跨帳戶存取權，您可以在其他帳戶內指定所有帳戶或 IAM 實體作為資源型政策的主體。如需詳細資訊，請參閱《IAM 使用者指南》中的[IAM 中的快帳戶資源存取](#)。

EMR Serverless 的政策動作

支援政策動作：是

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

JSON 政策的 Action 元素描述您可以用來允許或拒絕政策中存取的動作。政策會使用動作來授予執行相關聯動作的許可。

若要參考 EMR Serverless 動作的清單，請參閱《服務授權參考》中的[Amazon EMR Serverless 的動作、資源和條件索引鍵](#)。

EMR Serverless 中的政策動作在動作之前使用以下字首。

```
emr-serverless
```

若要在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [  
  "emr-serverless:action1",  
  "emr-serverless:action2"  
]
```

若要存取 Amazon EMR Serverless 身分型政策的範例，請參閱 [EMR Serverless 的身分型政策範例](#)。

EMR Serverless 的政策資源

支援政策資源：是

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

Resource JSON 政策元素可指定要套用動作的物件。最佳實務是使用其 [Amazon Resource Name \(ARN\)](#) 來指定資源。若動作不支援資源層級許可，使用萬用字元 (*) 表示該陳述式適用於所有資源。

```
"Resource": "*"
```

若要參考 Amazon EMR Serverless 資源類型及其 ARNs 的清單，請參閱《服務授權參考》中的 [Amazon EMR Serverless 定義的資源](#)。若要了解哪些動作指定每個資源的 ARN，請參閱 [Amazon EMR Serverless 的動作、資源和條件索引鍵](#)。

若要存取 Amazon EMR Serverless 身分型政策的範例，請參閱 [EMR Serverless 的身分型政策範例](#)。

EMR Serverless 的政策條件索引鍵

政策條件金鑰支援

支援服務特定政策條件金鑰	否
--------------	---

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

Condition 元素會根據定義的條件，指定陳述式的執行時機。您可以建立使用[條件運算子](#)的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。若要查看所有 AWS 全域條件索引鍵，請參閱《IAM 使用者指南》中的[AWS 全域條件內容索引鍵](#)。

若要參考 Amazon EMR Serverless 條件金鑰清單，並了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》中的 [Amazon EMR Serverless 的動作、資源和條件金鑰](#)。

所有 Amazon EC2 操作都支援 `aws:RequestedRegion` 和 `ec2:Region` 條件索引鍵。如需詳細資訊，請參閱[範例：限制對特定區域的存取](#)。

EMR Serverless 中的存取控制清單 (ACLs)

支援 ACL：否

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

使用 EMR Serverless 的屬性型存取控制 (ABAC)

屬性型存取控制 (ABAC) 支援

支援 ABAC (政策中的標籤)	是
------------------	---

屬性型存取控制 (ABAC) 是一種授權策略，依據稱為標籤的屬性來定義許可。您可以將標籤連接至 IAM 實體 AWS 和資源，然後設計 ABAC 政策，以便在委託人的標籤符合資源上的標籤時允許操作。

如需根據標籤控制存取，請使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 條件索引鍵，在政策的[條件元素](#)中，提供標籤資訊。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱《IAM 使用者指南》中的[使用 ABAC 授權定義許可](#)。如要查看含有設定 ABAC 步驟的教學課程，請參閱《IAM 使用者指南》中的[使用屬性型存取控制 \(ABAC\)](#)。

搭配 EMR Serverless 使用臨時登入資料

支援臨時憑證：是

臨時登入資料提供 AWS 資源的短期存取權，並在您使用聯合或切換角色時自動建立。AWS 建議您動態產生臨時登入資料，而不是使用長期存取金鑰。如需詳細資訊，請參閱《IAM 使用者指南》中的 [IAM 中的臨時安全憑證與可與 IAM 搭配運作的 AWS 服務](#)。

EMR Serverless 的跨服務主體許可

支援轉寄存取工作階段 (FAS)：是

轉送存取工作階段 (FAS) 使用呼叫的委託人許可 AWS 服務，並結合 AWS 服務請求向下游服務提出請求。如需提出 FAS 請求時的政策詳細資訊，請參閱 [轉發存取工作階段](#)。

EMR Serverless 的服務角色

支援服務角色	否
--------	---

EMR Serverless 的服務連結角色

支援服務連結角色	是
----------	---

如需建立或管理服務連結角色的詳細資訊，請參閱 [AWS 使用 IAM 的服務](#)。在資料表中尋找服務，其中包含服務連結角色欄中的 Yes。選擇是連結以存取該服務的服務連結角色文件。

使用 EMR Serverless 的服務連結角色

Amazon EMR Serverless 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 EMR Serverless 的唯一 IAM 角色類型。服務連結角色是由 EMR Serverless 預先定義，並包含該服務代表您呼叫其他 AWS 服務所需的所有許可。

服務連結角色可讓您更輕鬆地設定 EMR Serverless，因為您不必手動新增必要的許可。EMR Serverless 定義其服務連結角色的許可，除非另有定義，否則只有 EMR Serverless 可以擔任其角色。定義的許可包括信任政策和許可政策，且該許可政策無法附加至其他 IAM 實體。

您必須先刪除服務連結角色的相關資源，才能將其刪除。這可保護您的 EMR Serverless 資源，因為您不會不小心移除存取資源的許可。

如需有關支援服務連結角色的其他服務的資訊，請參閱 [AWS 使用 IAM 的服務](#)，並在服務連結角色欄中檢查是否有是的服務。選擇有連結的是，以存取該服務的服務連結角色文件。

EMR Serverless 的服務連結角色許可

EMR Serverless 使用名為 `AWSServiceRoleForAmazonEMRServerless` 的服務連結角色，讓它代表您呼叫 AWS APIs。

`AWSServiceRoleForAmazonEMRServerless` 服務連結角色信任下列服務擔任該角色：

- `ops.emr-serverless.amazonaws.com`

名為 `AmazonEMRServerlessServiceRolePolicy` 的角色許可政策允許 EMR Serverless 對指定的資源完成下列動作。

Note

受管政策內容會變更，因此此處顯示的政策可能已過期。在 [中檢視up-to-date政策](#) `AmazonEMRServerlessServiceRolePolicy` AWS 管理主控台。

- 動作：`ec2:CreateNetworkInterface`
- 動作：`ec2>DeleteNetworkInterface`
- 動作：`ec2:DescribeNetworkInterfaces`
- 動作：`ec2:DescribeSecurityGroups`
- 動作：`ec2:DescribeSubnets`
- 動作：`ec2:DescribeVpcs`
- 動作：`ec2:DescribeDhcpOptions`
- 動作：`ec2:DescribeRouteTables`
- 動作：`cloudwatch:PutMetricData`

以下是完整的`AmazonEMRServerlessServiceRolePolicy`政策。

JSON

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

```

    "Sid": "EC2PolicyStatement",
    "Effect": "Allow",
    "Action": [
      "ec2:CreateNetworkInterface",
      "ec2:DeleteNetworkInterface",
      "ec2:DescribeNetworkInterfaces",
      "ec2:DescribeSecurityGroups",
      "ec2:DescribeSubnets",
      "ec2:DescribeVpcs",
      "ec2:DescribeDhcpOptions",
      "ec2:DescribeRouteTables"
    ],
    "Resource": [
      "*"
    ]
  },
  {
    "Sid": "CloudWatchPolicyStatement",
    "Effect": "Allow",
    "Action": [
      "cloudwatch:PutMetricData"
    ],
    "Resource": [
      "*"
    ],
    "Condition": {
      "StringEquals": {
        "cloudwatch:namespace": [
          "AWS/EMRServerless",
          "AWS/Usage"
        ]
      }
    }
  }
]
}

```

下列信任政策會連接到此角色，以允許 EMR Serverless 委託人擔任此角色。

JSON

```
{
```

```

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

```

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](#)。

為 EMR Serverless 建立服務連結角色

您不需要手動建立服務連結角色，當您在 AWS 管理主控台 (使用 EMR Studio) AWS CLI、或 API 中建立新的 EMR Serverless 應用程式時 AWS，EMR Serverless 會為您建立服務連結角色。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。

使用 IAM 建立 AWSServiceRoleForAmazonEMRServerless 服務連結角色

將下列陳述式新增至需要建立服務連結角色的 IAM 實體的許可政策。

```

{
  "Effect": "Allow",
  "Action": [
    "iam:CreateServiceLinkedRole"
  ],
  "Resource": "arn:aws:iam::*:role/aws-service-role/ops.emr-serverless.amazonaws.com/AWSServiceRoleForAmazonEMRServerless*",
  "Condition": {"StringLike": {"iam:AWSServiceName": "ops.emr-serverless.amazonaws.com"}}
}

```

如果您刪除此服務連結角色，然後需要再次建立，請使用相同的程序在帳戶中重新建立角色。當您建立新的 EMR Serverless 應用程式時，EMR Serverless 會再次為您建立服務連結角色。

您也可以使用 IAM 主控台建立具有 EMR Serverless 使用案例的服務連結角色。在 AWS CLI 或 AWS API 中，使用服務名稱建立 `ops.emr-serverless.amazonaws.com` 服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的[建立服務連結角色](#)。如果您刪除此服務連結角色，請使用相同的程序再次建立角色。

編輯 EMR Serverless 的服務連結角色

EMR Serverless 不允許您編輯 `AWSServiceRoleForAmazonEMRServerless` 服務連結角色，因為各種實體可能會參考角色。您無法編輯 EMR Serverless 服務連結角色使用的擁有 AWS IAM 政策，因為它包含 EMR Serverless 所需的所有必要許可。然而，您可使用 IAM 來編輯角色描述。

使用 IAM 編輯 `AWSServiceRoleForAmazonEMRServerless` 服務連結角色的描述

將下列陳述式新增至 IAM 實體編輯服務連結角色描述所需的許可政策：

```
{
  "Effect": "Allow",
  "Action": [
    "iam: UpdateRoleDescription"
  ],
  "Resource": "arn:aws:iam::*:role/aws-service-role/ops.emr-serverless.amazonaws.com/AWSServiceRoleForAmazonEMRServerless*",
  "Condition": {"StringLike": {"iam:AWSServiceName": "ops.emr-serverless.amazonaws.com"}}
}
```

如需詳細資訊，請參閱《IAM 使用者指南》中的[編輯服務連結角色](#)。

刪除 EMR Serverless 的服務連結角色

如果您不再需要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。這樣您就沒有未主動監控或維護的未使用實體。不過，請先刪除所有區域中的所有 EMR Serverless 應用程式，再刪除服務連結角色。

Note

如果您嘗試刪除與角色相關聯的資源時，EMR Serverless 服務正在使用該角色，則刪除可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試操作。

使用 IAM 刪除 `AWSServiceRoleForAmazonEMRServerless` 服務連結角色

將下列陳述式新增至需要刪除服務連結角色之 IAM 實體的許可政策。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:DeleteServiceLinkedRole",
    "iam:GetServiceLinkedRoleDeletionStatus"
  ],
  "Resource": "arn:aws:iam::*:role/aws-service-role/ops.emr-serverless.amazonaws.com/AWSServiceRoleForAmazonEMRServerless*",
  "Condition": {"StringLike": {"iam:AWSServiceName": "ops.emr-serverless.amazonaws.com"}}
}
```

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 AWSServiceRoleForAmazonEMRServerless 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](#)。

EMR Serverless 服務連結角色支援的 區域

EMR Serverless 支援在提供服務的所有區域中使用服務連結角色。如需詳細資訊，請參閱[AWS 區域和端點](#)。

Amazon EMR Serverless 的任務執行期角色

您可以指定 EMR Serverless 任務執行可在代您呼叫其他服務時擔任的 IAM 角色許可。這包括存取 Amazon S3 的任何資料來源、目標，以及其他 AWS 資源，例如 Amazon Redshift 叢集和 DynamoDB 資料表。若要進一步了解如何建立角色，請參閱[建立任務執行期角色](#)。

範例執行時間政策

您可以將執行期政策，例如下列政策連接至任務執行期角色。下列任務執行時間政策允許：

- 使用 EMR 範例讀取 Amazon S3 儲存貯體的存取權。
- 完整存取 S3 儲存貯體。
- 建立和讀取 Glue Data Catalog AWS 的存取權。

若要新增對其他資源的存取權，例如 DynamoDB，您需要在建立執行時間角色時，在政策中包含這些 AWS 資源的許可。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadAccessForEMRSamples",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3::*.elasticmapreduce",
        "arn:aws:s3::*.elasticmapreduce/*"
      ]
    },
    {
      "Sid": "FullAccessToS3Bucket",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3::amzn-s3-demo-bucket",
        "arn:aws:s3::amzn-s3-demo-bucket/*"
      ]
    },
    {
      "Sid": "GlueCreateAndReadDataCatalog",
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabase",
        "glue:CreateDatabase",
        "glue:GetDataBases",
        "glue:CreateTable",
        "glue:GetTable",
        "glue:UpdateTable",
        "glue>DeleteTable",
        "glue:GetTables",

```

```

    "glue:GetPartition",
    "glue:GetPartitions",
    "glue:CreatePartition",
    "glue:BatchCreatePartition",
    "glue:GetUserDefinedFunctions"
  ],
  "Resource": [
    "*"
  ]
}
]
}

```

傳遞角色權限

您可以將 IAM 許可政策連接至使用者的角色，以允許使用者只傳遞已核准的角色。這可讓管理員控制哪些使用者可以將特定任務執行期角色傳遞給 EMR Serverless 任務。若要進一步了解如何設定許可，請參閱[授予使用者將角色傳遞至 AWS 服務的許可](#)。

以下是允許將任務執行期角色傳遞給 EMR Serverless 服務主體的範例政策。

```

{
  "Effect": "Allow",
  "Action": "iam:PassRole",
  "Resource": "arn:aws:iam::1234567890:role/JobRuntimeRoleForEMRServerless",
  "Condition": {
    "StringLike": {
      "iam:PassedToService": "emr-serverless.amazonaws.com"
    }
  }
}

```

與執行期角色相關聯的受管許可政策

當您透過 EMR Studio 主控台將任務執行提交至 EMR Serverless 時，有一個步驟可讓您選擇要與應用程式建立關聯的執行期角色。主控台每個選擇都有相關的基礎受管政策，請務必注意這些政策。三個選項如下：

1. 所有儲存貯體 – 當您選擇此選項時，它會指定 [AmazonS3FullAccess](#) AWS 受管政策，提供所有儲存貯體的完整存取權。

2. 特定儲存貯體 – 這會指定您選擇的每個儲存貯體的 Amazon 資源名稱 (ARN) 識別符。不包含基礎受管政策。
3. 無 – 不包含受管政策許可。

我們建議新增特定儲存貯體。如果您選擇所有儲存貯體，請記住，它會設定所有儲存貯體的完整存取權。

EMR Serverless 的使用者存取政策範例

您可以根據希望每個使用者在與 EMR Serverless 應用程式互動時執行的動作，為使用者設定精細的政策。下列政策範例可能有助於為您的使用者設定適當的許可。本節僅著重於 EMR Serverless 政策。如需 EMR Studio 使用者政策的範例，請參閱[設定 EMR Studio 使用者許可](#)。如需如何將政策連接至 IAM 使用者（原則）的資訊，請參閱 [《IAM 使用者指南》中的管理 IAM 政策](#)。

進階使用者政策

若要授予 EMR Serverless 的所有必要動作，請建立 AmazonEMRServerlessFullAccess 政策並將其連接至所需的 IAM 使用者、角色或群組。

以下是範例政策，允許進階使用者建立和修改 EMR Serverless 應用程式，以及執行其他動作，例如提交和偵錯任務。它會顯示 EMR Serverless 對其他服務所需的所有動作。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessActions",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication",
        "emr-serverless:UpdateApplication",
        "emr-serverless>DeleteApplication",
        "emr-serverless:ListApplications",
        "emr-serverless:GetApplication",
        "emr-serverless:StartApplication",
        "emr-serverless:StopApplication",
        "emr-serverless:StartJobRun",
        "emr-serverless:CancelJobRun",

```

```

    "emr-serverless:ListJobRuns",
    "emr-serverless:GetJobRun"
  ],
  "Resource": [
    "*"
  ]
}
]
}

```

當您啟用 VPC 的網路連線時，EMR Serverless 應用程式會建立 Amazon EC2 彈性網路介面 (ENIs) 來與 VPC 資源通訊。下列政策可確保僅在 EMR Serverless 應用程式的內容中建立新的 EC2 ENIs。

Note

我們強烈建議設定此政策，以確保使用者無法建立 EC2 ENIs 除非在啟動 EMR Serverless 應用程式的情況下。

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowEC2ENICreationWithEMRTags",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:network-interface/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:CalledViaLast": "ops.emr-serverless.amazonaws.com"
        }
      }
    }
  ]
}

```

}

如果您想要限制 EMR Serverless 存取特定子網路，您可以使用標籤條件來標記每個子網路。此 IAM 政策可確保 EMR Serverless 應用程式只能在允許的子網路內建立 EC2 ENIs。

```
{
  "Sid": "AllowEC2ENICreationInSubnetAndSecurityGroupWithEMRTags",
  "Effect": "Allow",
  "Action": [
    "ec2:CreateNetworkInterface"
  ],
  "Resource": [
    "arn:aws:ec2:*:*:subnet/*",
    "arn:aws:ec2:*:*:security-group/*"
  ],
  "Condition": {
    "StringEquals": {
      "aws:ResourceTag/KEY": "VALUE"
    }
  }
}
```

Important

如果您是建立第一個應用程式的管理員或進階使用者，您必須設定許可政策，以允許您建立 EMR Serverless 服務連結角色。若要進一步了解，請參閱 [使用 EMR Serverless 的服務連結角色](#)。

下列 IAM 政策可讓您為帳戶建立 EMR Serverless 服務連結角色。

```
{
  "Sid": "AllowEMRServerlessServiceLinkedRoleCreation",
  "Effect": "Allow",
  "Action": "iam:CreateServiceLinkedRole",
  "Resource": "arn:aws:iam::account-id:role/aws-service-role/ops.emr-serverless.amazonaws.com/AWSServiceRoleForAmazonEMRServerless"
}
```

資料工程師政策

以下是允許使用者在 EMR Serverless 應用程式上唯讀許可的範例政策，以及提交和偵錯任務的功能。請切記，因為此政策並未明確拒絕動作，不同的政策陳述式仍有可能用於授予指定動作存取權。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EMRServerlessActions",
      "Effect": "Allow",
      "Action": [
        "emr-serverless:ListApplications",
        "emr-serverless:GetApplication",
        "emr-serverless:StartApplication",
        "emr-serverless:StartJobRun",
        "emr-serverless:CancelJobRun",
        "emr-serverless:ListJobRuns",
        "emr-serverless:GetJobRun"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

使用存取控制的標籤

您可以使用標籤條件進行精細存取控制。例如，您可以限制一個團隊的使用者，讓他們只能將任務提交至標示其團隊名稱的 EMR Serverless 應用程式。

JSON

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

```
"Sid": "EMRServerlessActions",
"Effect": "Allow",
"Action": [
  "emr-serverless:ListApplications",
  "emr-serverless:GetApplication",
  "emr-serverless:StartApplication",
  "emr-serverless:StartJobRun",
  "emr-serverless:CancelJobRun",
  "emr-serverless:ListJobRuns",
  "emr-serverless:GetJobRun"
],
"Resource": [
  "*"
]
}
]
```

標籤型存取控制的策略

您可以在身分型政策中使用條件，根據標籤控制對應用程式和任務執行的存取。

下列範例示範將條件運算子與 EMR Serverless 條件索引鍵搭配使用的不同案例和方法。這些 IAM 政策陳述式僅作示範用途，不應用於生產環境。有多種方法可以結合政策陳述式，以根據您的需求授予和拒絕許可。如需規劃和測試 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](#)。

以下範例示範用於控制 EMR Serverless 應用程式所允許之動作的身分型許可政策。

僅在具有特定標籤值的資源上允許動作

在下列政策範例中，StringEquals 條件運算子會嘗試 dev 符合標籤部門的值。如果標籤部門尚未新增至應用程式，或不包含值 dev，則政策不適用，且此政策不允許這些動作。如果沒有其他政策陳述式允許動作，則使用者只能使用具有此值之此標籤的應用程式。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:GetApplication"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/department": "dev"
        }
      },
      "Sid": "AllowEMRSERVERLESSGetapplication"
    }
  ]
}
```

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標籤包含值dev或的應用程式上允許動作test，請將先前範例中的條件區塊取代為以下內容。

```
"Condition": {
  "StringEquals": {
    "aws:ResourceTag/department": ["dev", "test"]
  }
}
```

建立資源時需要進行標記

在下面的範例中，建立應用程式時需要套用標籤。

JSON

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

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "emr-serverless:CreateApplication"
    ],
    "Resource": [
      "*"
    ],
    "Condition": {
      "StringEquals": {
        "aws:RequestedRegion": "us-east-1"
      }
    },
    "Sid": "AllowEMRSERVERLESSCreateapplication"
  }
]
}

```

下列政策陳述式僅允許使用者在應用程式具有標籤時建立應用程式，該department標籤可包含任何值。

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": ["us-east-1", "us-west-2"]
        }
      },
      "Sid": "AllowEMRSERVERLESSCreateapplication"
    }
  ]
}

```

```
    }  
  ]  
}
```

拒絕新增和移除標籤的存取權

此政策可防止使用者新增或移除 EMR Serverless 應用程式上的標籤，其 department 標籤值不是 dev。

JSON

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

EMR Serverless 的身分型政策範例

根據預設，使用者和角色沒有建立或修改 Amazon EMR Serverless 資源的許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。

如需了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的[建立 IAM 政策 \(主控台\)](#)。

如需 Amazon EMR Serverless 定義之動作和資源類型的詳細資訊，包括每種資源類型的 ARNs 格式，請參閱《服務授權參考》中的 [Amazon EMR Serverless 的動作、資源和條件索引鍵](#)。

主題

- [政策最佳實務](#)
- [允許使用者存取自己的許可](#)

政策最佳實務

Note

EMR Serverless 不支援受管政策，因此下列第一個做法不適用。

身分型政策會判斷您帳戶中的某個人員是否可以建立、存取或刪除 Amazon EMR Serverless 資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管政策並轉向最低權限許可 – 若要開始將許可授予您的使用者和工作負載，請使用將許可授予許多常見使用案例的 AWS 受管政策。它們可在您的 中使用 AWS 帳戶。我們建議您定義特定於使用案例 AWS 的客戶受管政策，以進一步減少許可。如需更多資訊，請參閱《IAM 使用者指南》中的 [AWS 受管政策](#) 或 [任務職能的 AWS 受管政策](#)。
- 套用最低權限許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的許可。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱《IAM 使用者指南》中的 [IAM 中的政策和許可](#)。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。如果透過特定 等使用服務動作 AWS 服務，您也可以使用條件來授予其存取權 CloudFormation。如需詳細資訊，請參閱《IAM 使用者指南》中的 [IAM JSON 政策元素：條件](#)。
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您撰寫安全且實用的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的 [使用 IAM Access Analyzer 驗證政策](#)。
- 需要多重要素驗證 (MFA) – 如果您的案例需要 IAM 使用者或 中的根使用者 AWS 帳戶，請開啟 MFA 以提高安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的 [透過 MFA 的安全 API 存取](#)。

如需 IAM 中最佳實務的相關資訊，請參閱《IAM 使用者指南》中的 [IAM 安全最佳實務](#)。

允許使用者存取自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視連接到他們使用者身分的內嵌及受管政策。此政策包含在主控台或使用或 AWS CLI AWS API 以程式設計方式完成此動作的許可。

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

AWS 受管政策的 Amazon EMR Serverless 更新

自此服務開始追蹤這些變更以來，存取 Amazon EMR Serverless AWS 受管政策更新的詳細資訊。如需此頁面變更的自動提醒，請訂閱 Amazon EMR Serverless [文件歷史記錄](#) 頁面上的 RSS 摘要。

變更	描述	Date
AmazonEMRServerlessServiceRolePolicy – 更新至現有政策	Amazon EMR Serverless 將新的 SidCloudWatchPolicyStatement 和 EC2PolicyStatement 新增至 AmazonEMRServerlessServiceRolePolicy 政策 。	2024 年 1 月 25 日
AmazonEMRServerlessServiceRolePolicy – 更新至現有政策	Amazon EMR Serverless 新增了新的許可，以允許 Amazon EMR Serverless 發佈 "AWS/Usage" 命名空間中 vCPU 用量的彙總帳戶指標。	2023 年 4 月 20 日
Amazon EMR Serverless 已開始追蹤變更	Amazon EMR Serverless 開始追蹤其 AWS 受管政策的變更。	2023 年 4 月 20 日

對 Amazon EMR Serverless 身分和存取進行故障診斷

使用以下資訊來協助您診斷和修正使用 Amazon EMR Serverless 和 IAM 時可能遇到的常見問題。

主題

- [我無權在 Amazon EMR Serverless 中執行動作](#)
- [我未獲得執行 iam:PassRole 的授權](#)
- [我想要允許 AWS 帳戶外的人員存取我的 Amazon EMR Serverless 資源](#)
- [我無法從 EMR Studio 開啟即時 UI/Spark 歷史記錄伺服器來偵錯我的任務，或當我嘗試使用 取得日誌時發生 API 錯誤 get-dashboard-for-job-run](#)

我無權在 Amazon EMR Serverless 中執行動作

如果 AWS 管理主控台告知您無權執行動作，請聯絡您的管理員尋求協助。您的管理員是為您提供使用者名稱和密碼的人員。

當mateojackson使用者嘗試使用主控台存取虛構`my-example-widget`資源的詳細資訊，但沒有虛構`emr-serverless:GetWidget`許可時，會發生下列範例錯誤。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: emr-serverless: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-serverless:GetWidget` 資源。

我未獲得執行 iam:PassRole 的授權

如果您收到錯誤，告知您無權執行 `iam:PassRole` 動作，您的政策必須更新，以允許您將角色傳遞給 Amazon EMR Serverless。

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

當名為 `marymajor` 的 IAM 使用者嘗試使用主控台在 Amazon EMR Serverless 中執行動作時，會發生下列範例錯誤。但是，動作請求服務具備服務角色授予的許可。Mary 沒有將角色傳遞給服務的許可。

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

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

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我想要允許 AWS 帳戶外的人員存取我的 Amazon EMR Serverless 資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要了解 Amazon EMR Serverless 是否支援這些功能，請參閱 [Amazon EMR Serverless 中的 Identity and Access Management \(IAM\)](#)。
- 若要了解如何提供您擁有 AWS 帳戶的資源存取權，請參閱 [《IAM 使用者指南》中的在您擁有 AWS 帳戶的另一個中為 IAM 使用者提供存取權](#)。
- 若要了解如何將資源的存取權提供給第三方 AWS 帳戶，請參閱 [《IAM 使用者指南》中的將存取權提供給第三方 AWS 帳戶擁有](#)。
- 如需了解如何透過聯合身分提供存取權，請參閱 [《IAM 使用者指南》中的將存取權提供給在外部進行身分驗證的使用者 \(聯合身分\)](#)。
- 如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱 [《IAM 使用者指南》中的 IAM 中的跨帳戶資源存取](#)。

我無法從 EMR Studio 開啟即時 UI/Spark 歷史記錄伺服器來偵錯我的任務，或當我嘗試使用取得日誌時發生 API 錯誤 `get-dashboard-for-job-run`

如果您使用 EMR Serverless 受管儲存體進行記錄，且 EMR Serverless 應用程式位於具有 Amazon S3 VPC 端點的私有子網路中，且您連接端點政策來控制存取，請將在 VPC 政策中使用 [受管儲存體的 EMR Serverless 記錄](#) 中所述的許可新增至 S3 閘道端點，供 EMR Serverless 存放和提供應用程式日誌。

受信任的身分傳播

透過 Amazon EMR 7.8.0 版及更高版本，您可以透過 Apache Livy 端點使用 EMR Serverless 將使用者身分從 AWS IAM Identity Center 傳播到互動式工作負載。Apache Livy 互動式工作負載將進一步將提供的身分傳播到下游服務，例如 Amazon S3、Lake Formation 和 Amazon Redshift，透過這些下游的使用者身分啟用安全的資料存取。下列各節提供透過 Apache Livy 端點使用 EMR Serverless 啟動和傳播身分至互動式工作負載所需的概觀、先決條件和步驟。

概觀

對於任何大小和類型的組織，[IAM Identity Center](#) 是建議在 AWS 進行人力資源身分驗證和授權的方法。使用 Identity Center，在中建立和管理使用者身分 AWS，或連接現有的身分來源，包括 Microsoft Active Directory、Okta、Ping Identity、JumpCloud、Google Workspace 和 Microsoft Entra ID (先前稱為 Azure AD)。

[信任的身分傳播](#) 是一種 AWS IAM Identity Center 功能，連線 AWS 服務的管理員可以使用此功能來授予和稽核服務資料的存取權。存取此資料是根據使用者屬性，例如群組關聯。設定信任的身分傳播需要

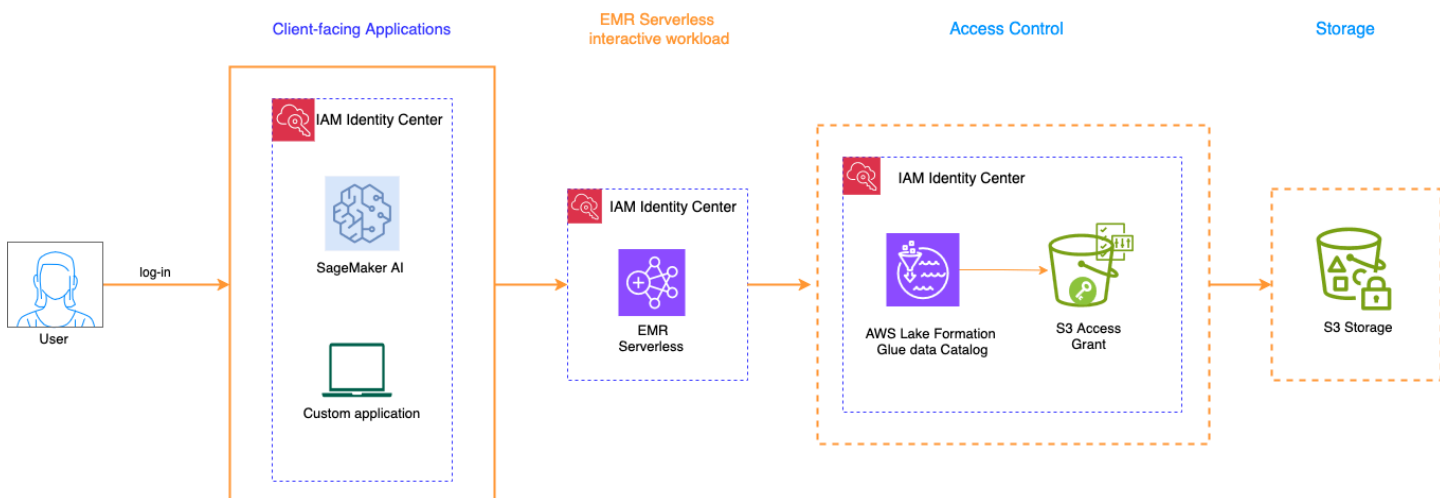
連線 AWS 服務的管理員與 IAM Identity Center 管理員之間的協同合作。如需詳細資訊，請參閱《IAM Identity Center 使用者指南》中的[先決條件和考量](#)事項。

功能和優勢

EMR Serverless Apache Livy 端點與 IAM Identity Center [Trusted Identity Propagation](#) 整合可提供下列優點：

- 能夠對 AWS Lake Formation 受管 Glue AWS 資料目錄資料表上的 Identity Center 身分強制執行資料表層級授權。
- 能夠在 Amazon Redshift 叢集上使用 Identity Center 身分強制執行授權。
- 啟用使用者動作的端對端追蹤以進行稽核。
- 能夠在 S3 Access Grants 受管 S3 字首上使用 Identity Center 身分，強制執行 Amazon S3 字首層級授權。

運作方式



使用案例範例

資料準備與特徵工程

來自多個研究團隊的資料科學家使用統一的資料平台來協作處理複雜的專案。他們使用其公司登入資料登入 SageMaker AI，立即存取跨越多個 AWS 帳戶的大量共用資料湖。當他們開始為新的機器學習模型進行功能工程時，透過 EMR Serverless 啟動的 Spark 工作階段會根據其傳播的身分強制執行 Lake Formation 的資料欄和資料列層級安全政策。科學家可以使用熟悉的工具有效率地準備資料和設計功能，而合規團隊可以確保每次資料互動都會自動追蹤和稽核。這種安全、協作的環境可加速研究管道，同時維持受監管產業所需的嚴格資料保護標準。

Trusted-Identity 傳播入門

本節可協助您使用 Apache Livy 端點設定 EMR-Serverless 應用程式，以與 AWS IAM Identity Center 整合，並啟用[受信任身分傳播](#)。

先決條件

- 您要在 AWS 區域中建立啟用受信任身分傳播的 EMR Serverless Apache Livy 端點的 Identity Center 執行個體。Identity Center 執行個體只能存在於 AWS 帳戶的單一區域中。請參閱[啟用 IAM Identity Center](#) 和[將身分來源中的使用者和群組佈建到 IAM Identity Center](#)。
- 為 Lake Formation 或 S3 Access Grants 或 Amazon Redshift 叢集等下游服務啟用受信任身分傳播，互動式工作負載會與之互動以存取資料。

建立啟用受信任身分傳播的 EMR Serverless 應用程式的許可

除了[存取 EMR Serverless 所需的基本許可](#)之外，您還必須為 IAM 身分或角色設定其他許可，用於建立啟用受信任身分傳播的 EMR Serverless 應用程式。對於受信任身分傳播，EMR Serverless 會在您的帳戶中建立/引導單一服務受管身分中心應用程式，該應用程式會利用該應用程式將身分驗證和身分傳播到下游。

```
"sso:DescribeInstance",  
"sso:CreateApplication",  
"sso>DeleteApplication",  
"sso:PutApplicationAuthenticationMethod",  
"sso:PutApplicationAssignmentConfiguration",  
"sso:PutApplicationGrant",  
"sso:PutApplicationAccessScope"
```

- `sso:DescribeInstance` – 准許描述和驗證您在 `identity-center-configuration` 參數中指定的 IAM Identity Center instanceArn。
- `sso:CreateApplication` – 准許建立用於 `trusted-identity-propatgion` 動作的 EMR Serverless 受管 IAM Identity Center 應用程式。
- `sso>DeleteApplication` – 准許清除 EMR Serverless 受管 IAM Identity Center 應用程式
- `sso:PutApplicationAuthenticationMethod` – 准許在 EMR Serverless 受管 IAM Identity Center 應用程式上放置 `authenticationMethod`，以允許 `emr-serverless` 服務主體與 IAM Identity Center 應用程式互動。

- `sso:PutApplicationAssignmentConfiguration` – 准許在 IAM Identity Center 應用程式上設定「User-assignment-not-required」設定。
- `sso:PutApplicationGrant` – 准許在 IAM Identity Center 應用程式上套用 `Token-exchange`、`introspectToken`、`refreshToken` 和 `revokeToken` 授予。
- `sso:PutApplicationAccessScope` – 准許將啟用信任身分傳播的下游範圍套用至 IAM Identity Center 應用程式。我們套用「`redshift:connect`」、「`lakeformation:query`」和「`s3:read_write`」範圍，以啟用這些服務的 `trusted-identity-propagation`。

建立啟用信任身分傳播的 EMR Serverless 應用程式

您必須使用指定 `--identity-center-configuration` 欄位 `identityCenterInstanceArn`，才能在應用程式中啟用信任身分傳播。使用以下範例命令建立已啟用信任身分傳播的 EMR Serverless 應用程式。

Note

您也必須將指定 `--interactive-configuration` `'{"livyEndpointEnabled":true}'` 為僅針對 Apache Livy 端點啟用受信任身分傳播。

```
aws emr-serverless create-application \  
  --release-label emr-7.8.0 \  
  --type "SPARK" \  
  --identity-center-configuration '{"identityCenterInstanceArn" :  
"arn:aws:sso:::instance/ssoins-123456789"}' \  
  --interactive-configuration '{"livyEndpointEnabled":true}'
```

- `identity-center-configuration` – (選用) 指定時啟用 Identity Center 信任的身分傳播。
- `identityCenterInstanceArn` : (必要) Identity Center 執行個體 ARN。

如果您沒有必要的 Identity Center 許可 (先前提及)，請先建立沒有信任身分傳播的 EMR Serverless 應用程式 (例如，不要指定 `--identity-center-configuration` 參數)，然後要求 Identity Center Admin 透過叫用更新應用程式 API 來啟用信任身分傳播，請參閱下列範例：

```
aws emr-serverless update-application \  
  --application-id applicationId \  
  --identity-center-configuration '{"identityCenterInstanceArn" :  
"arn:aws:sso:::instance/ssoins-123456789"}'
```

```
--identity-center-configuration '{"identityCenterInstanceArn" :  
"arn:aws:sso:::instance/ssoins-123456789"}'
```

EMR Serverless 會在您的帳戶中建立服務受管身分中心應用程式，服務會利用該應用程式來驗證身分，並將身分傳播到下游服務。EMR Serverless 建立的受管 Identity Center 應用程式會與您帳戶中所有啟用trusted-identity-propagation的 EMR Serverless 應用程式共用。

Note

請勿手動修改受管 Identity Center 應用程式上的設定。任何變更都可能影響您帳戶中所有啟用trusted-identity-propagation的 EMR Serverless 應用程式。

任務執行角色傳播身分的許可

由於 EMR-Serverless 利用 Identity-enhanced job-execution-role 登入資料將身分傳播到下游 AWS 服務，因此 Job Execution Role 的 Trust-policy 必須具有額外的許可sts:SetContext，以使用身分增強任務執行角色登入資料，以允許受trusted-identity-propagation到下游服務，例如 S3 Access-grant、Lake Formation 或 Amazon Redshift。若要進一步了解如何建立角色，請參閱[建立任務執行期角色](#)。

JSON

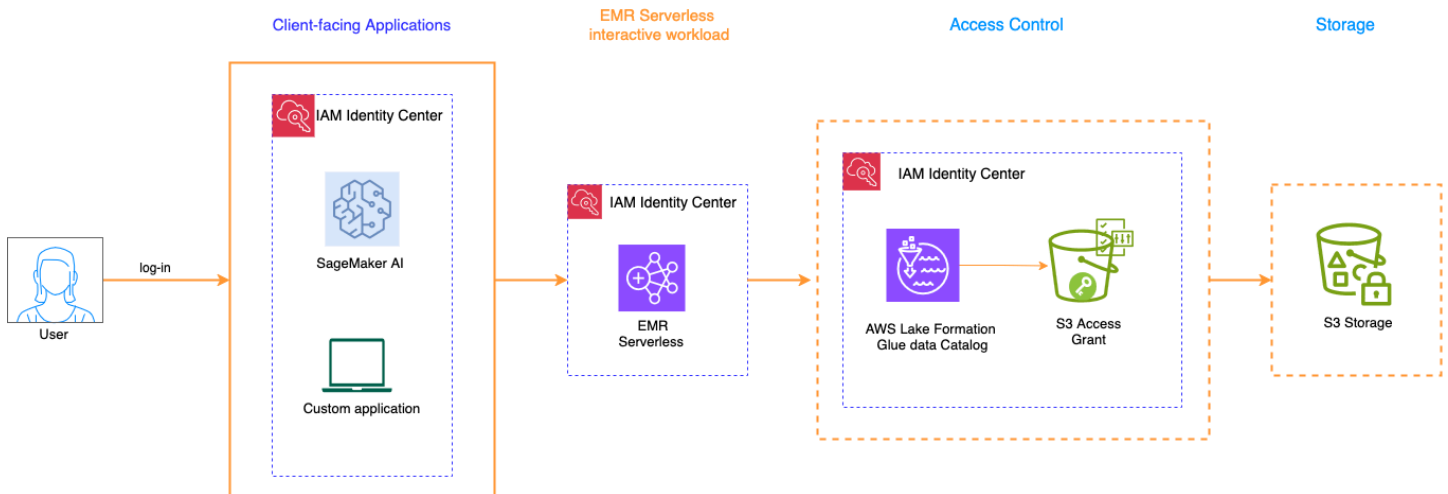
```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": {  
        "Service": "emr-serverless.amazonaws.com"  
      },  
      "Action": [ "sts:AssumeRole", "sts:SetContext"  
    ]  
  ]  
}
```

此外，JobExecutionRole 需要下游 AWS 服務的許可，其任務執行會使用使用者身分來擷取資料。請參閱以下連結來設定 S3 Access Grant、Lake Formation。

- [搭配 EMR Serverless 使用 Lake Formation](#)
- [搭配 EMR Serverless 使用 Amazon S3 Access Grants](#)

互動式工作負載的受信任身分傳播

透過 Apache Livy 端點將身分傳播到互動式工作負載的步驟，取決於您的使用者是與 AWS 受管開發環境互動，例如，Amazon SageMaker AI 還是您自己的自我託管筆記本環境，做為面向用戶端的應用程式。



AWS 受管開發環境

下列 AWS 受管用戶端面向應用程式支援使用 EMR-Serverless Apache Livy 端點的受信任身分傳播：

- [Amazon SageMaker AI](#)

客戶管理的自我託管筆記本環境

若要為自訂開發應用程式的使用者啟用受信任身分傳播，請參閱 AWS 安全部落格中的[使用受信任身分傳播以程式設計方式存取 AWS 服務](#)。

使用者背景工作階段

使用者背景工作階段可讓長時間執行的分析和機器學習流程繼續，即使使用者已從筆記本界面登出也一樣。此功能是透過 EMR Serverless 與 IAM Identity Center 受信任身分傳播功能的整合來實作。本節說明使用者背景工作階段的組態選項和行為。

Note

使用者背景工作階段適用於透過筆記本界面啟動的 Spark 工作負載，例如 Amazon SageMaker Unified Studio。啟用或停用此功能只會影響新的 Livy 工作階段；現有的作用中 Livy 工作階段不會受到影響。

設定使用者背景工作階段

使用者背景工作階段必須在兩個層級啟用，才能正常運作：

1. IAM Identity Center 執行個體層級 – 通常由 IdC 管理員設定
2. EMR Serverless 應用程式層級 – 由 EMR Serverless 應用程式管理員設定

為 EMR Serverless 應用程式啟用使用者背景工作階段

若要啟用 EMR Serverless 應用程式的使用者背景工作階段，您必須在建立或更新應用程式 `identityCenterConfiguration` 時，在 `true` 中將 `userBackgroundSessionsEnabled` 參數設定為。

先決條件

- 用於建立/更新 EMR Serverless 應用程式的 IAM 角色必須具有 `sso:PutApplicationSessionConfiguration` 許可。此許可允許 EMR Serverless 在 EMR Serverless 受管 IdC 應用程式層級啟用使用者背景工作階段。
- 您的 EMR Serverless 應用程式必須使用發行標籤 7.8 或更新版本，且必須啟用受信任身分傳播。

使用 啟用使用者背景工作階段 AWS CLI

```
aws emr-serverless create-application \  
  --name "my-analytics-app" \  
  --type "SPARK" \  
  --release-label "emr-7.8.0" \  
  --identity-center-configuration '{"identityCenterInstanceArn":  
"arn:aws:sso:::instance/ssoins-1234567890abcdef", "userBackgroundSessionsEnabled":  
true}'
```

若要更新現有的應用程式：

```
aws emr-serverless update-application \
  --application-id applicationId \
  --identity-center-configuration '{"identityCenterInstanceArn":
  "arn:aws:sso:::instance/ssoins-1234567890abcdef", "userBackgroundSessionsEnabled":
  true}'
```

組態矩陣

有效的使用者背景工作階段組態取決於 EMR Serverless 應用程式設定和 IAM Identity Center 執行個體層級設定：

使用者背景工作階段組態矩陣

IAM Identity Center userBackgroundSession 已啟用	EMR Serverless userBackgroundSessionsEnabled	Behavior (行為)
是	TRUE	已啟用使用者背景工作階段
是	FALSE	工作階段會隨著使用者登出而過期
否	TRUE	應用程式建立/更新失敗並出現例外狀況
否	FALSE	工作階段會隨著使用者登出而過期

預設使用者背景工作階段持續時間

根據預設，所有使用者背景工作階段在 IAM Identity Center 中的持續時間限制為 7 天。管理員可以在 IAM Identity Center 主控台中修改此持續時間。此設定適用於 IAM Identity Center 執行個體層級，影響該執行個體內所有支援的 IAM Identity Center 應用程式。

- 持續時間可以設定為 15 分鐘到 90 天之間的任何值。
- 此設定是在 IAM Identity Center 主控台的設定 → 身分驗證 → 設定（非互動式任務區段）下進行設定

Note

EMR Serverless Livy 工作階段有單獨的持續時間上限 24 小時。工作階段會在達到 Livy 工作階段限制或使用者背景工作階段持續時間時終止，以先到者為準。

停用使用者背景工作階段的影響

在 IAM Identity Center 中停用使用者背景工作階段時：

現有的 Livy 工作階段

如果啟動時已啟用使用者背景工作階段，請繼續執行而不會中斷。這些工作階段將繼續使用其現有的背景工作階段字串，直到其自然終止或明確停止為止。

新的 Livy 工作階段

將使用標準信任的身分傳播流程，並在使用者登出或其互動式工作階段過期時終止（例如關閉 Amazon SageMaker Unified Studio JupyterLab 筆記本時）。

變更使用者背景工作階段持續時間

在 IAM Identity Center 中修改使用者背景工作階段的持續時間設定時：

現有的 Livy 工作階段

繼續執行與啟動時相同的背景工作階段持續時間。

新的 Livy 工作階段

將為背景工作階段使用新的工作階段持續時間。

考量事項

工作階段終止條件

使用使用者背景工作階段時，Livy 工作階段會繼續執行，直到發生下列其中一種情況：

- 使用者背景工作階段過期（根據 IdC 組態，最多 90 天）
- 管理員會手動撤銷使用者背景工作階段
- Livy 工作階段達到閒置逾時（預設值：上次執行陳述式後 1 小時）

- Livy 工作階段達到其最長持續時間 (24 小時)
- 使用者明確停止或重新啟動筆記本核心

資料持久性

使用使用者背景工作階段時：

- 使用者一旦登出，就無法重新連線至其筆記本介面以檢視結果
- 設定您的 Spark 陳述式，在執行完成之前將結果寫入持久性儲存體（例如 Amazon S3）

成本影響

- 即使使用者結束其 Amazon SageMaker Unified Studio JupyterLab 工作階段，任務仍會繼續執行至完成，並在整個執行期間產生費用。
- 監控您的作用中背景工作階段，以避免忘記或放棄的工作階段產生不必要的成本。

功能可用性

EMR Serverless 的使用者背景工作階段可用於：

- 僅限 Spark 引擎（不支援 Hive 引擎）
- 僅限 Livy 互動式工作階段（不支援批次任務和串流任務）
- EMR Serverless 發行標籤 7.8 及更新版本

EMR Serverless Trusted-Identity-Propagation 整合的考量事項

當您將 IAM Identity Center Trusted-Identity-Propagation 與 EMR Serverless 應用程式搭配使用時，請考慮下列事項：

- Amazon EMR 7.8.0 及更高版本支援透過 Identity Center 的受信任身分傳播，且僅適用於 Apache Spark。
- 信任的身分傳播只能用於[透過 Apache Livy 端點搭配 EMR Serverless 的互動式工作負載](#)。透過 EMR Studio 的互動式工作負載不支援受信任身分傳播
- 批次任務和串流工作負載不支援受信任身分傳播
- 使用受信任身分傳播的 AWS Lake Formation 的精細存取控制適用於[透過 Apache Livy 端點使用 EMR Serverless 的互動式工作負載](#)。

- 下列 AWS 區域支援使用 Amazon EMR 的受信任身分傳播：
 - af-south-1 – 非洲 (開普敦)
 - ap-east-1 – 亞太區域 (香港)
 - ap-northeast-1 – 亞太區域 (東京)
 - ap-northeast-2 – 亞太區域 (首爾)
 - ap-northeast-3 – 亞太區域 (大阪)
 - ap-south-1 – 亞太區域 (孟買)
 - ap-southeast-1 – 亞太區域 (新加坡)
 - ap-southeast-2 – 亞太區域 (雪梨)
 - ap-southeast-3 – 亞太區域 (雅加達)
 - ca-central-1 – 加拿大 (中部)
 - ca-west-1 – 加拿大 (卡加利)
 - eu-central-1 – 歐洲 (法蘭克福)
 - eu-north-1 – 歐洲 (斯德哥爾摩)
 - eu-south-1 – 歐洲 (米蘭)
 - eu-south-2 – 歐洲 (西班牙)
 - eu-west-1 – 歐洲 (愛爾蘭)
 - eu-west-2 – 歐洲 (倫敦)
 - eu-west-3 – 歐洲 (巴黎)
 - me-central-1 – 中東 (阿拉伯聯合大公國)
 - me-south-1 – 中東 (巴林)
 - sa-east-1 – 南美洲 (聖保羅)
 - us-east-1 – 美國東部 (維吉尼亞北部)
 - us-east-2 – 美國東部 (俄亥俄州)
 - us-west-1 – 美國西部 (加利佛尼亞北部)
 - us-west-2 – 美國西部 (奧勒岡州)

搭配 EMR Serverless 使用 Lake Formation

您可以將 EMR Serverless 應用程式設定為搭配完整資料表存取或精細存取控制使用 Lake Formation。²⁷⁰ 如需每個存取模式中支援功能的詳細資訊，請檢閱下表。

功能可用性

功能	可從 取得
Hive、Iceberg 資料表的讀取操作 (SELECT、DESCRIBE)	EMR 7.2+
多向檢視	EMR 7.6+
Delta Lake 和 Hudi 資料表的讀取操作 (SELECT、DESCRIBE)	EMR 7.6+
Hive、Iceberg 的完整資料表存取	EMR 7.9+
Delta Lake 的完整資料表存取	EMR 7.11+
Hive、Iceberg 和 Delta Lake 資料表的寫入操作 (DDL、DML)	EMR 7.12+
Hudi 的完整資料表存取	EMR 7.12+

EMR Serverless 的 Lake Formation 完整資料表存取

使用 Amazon EMR 7.8.0 版及更高版本，您可以利用 AWS Lake Formation 搭配 Glue Data Catalog，其中任務執行期角色具有完整的資料表許可，而不受精細存取控制的限制。此功能可讓您從 EMR Serverless Spark 批次和互動式任務讀取和寫入受 Lake Formation 保護的資料表。請參閱下列各節，進一步了解 Lake Formation 以及如何搭配 EMR Serverless 使用。

使用 Lake Formation 搭配完整資料表存取

您可以從 EMR Serverless Spark 任務或互動式工作階段存取 AWS Lake Formation 保護的 Glue Data 目錄資料表，其中任務的執行期角色具有完整的資料表存取權。您不需要在 EMR Serverless 應用程式上啟用 AWS Lake Formation。當 Spark 任務設定為完整資料表存取 (FTA) 時，AWS Lake Formation 登入資料用於讀取/寫入 AWS Lake Formation 註冊資料表的 S3 資料，而任務的執行時間角色登入資料將用於讀取/寫入未向 AWS Lake Formation 註冊的資料表。

⚠ Important

請勿為精細存取控制啟用 AWS Lake Formation。任務無法同時在相同的 EMR 叢集或應用程式上執行完整資料表存取 (FTA) 和精細存取控制 (FGAC)。

步驟 1：在 Lake Formation 中啟用完整資料表存取

若要使用完整資料表存取 (FTA) 模式，您必須允許第三方查詢引擎存取資料，而不需要 AWS Lake Formation 中的 IAM 工作階段標籤驗證。若要啟用，請遵循[整合應用程式進行完整資料表存取](#)中的步驟。

📘 Note

存取跨帳戶資料表時，必須在生產者和消費者帳戶中啟用完整資料表存取。同樣地，存取跨區域資料表時，必須在生產者和消費者區域啟用此設定。

步驟 2：設定任務執行時期角色的 IAM 許可

對於基礎資料的讀取或寫入存取權，除了 Lake Formation 許可之外，任務執行期角色還需要 IAM `lakeformation:GetDataAccess` 許可。有了此許可，Lake Formation 就會授與要求存取資料所需的臨時憑證。

以下範例政策說明如何提供 IAM 許可，以存取 Amazon S3 中的指令碼、將日誌上傳至 S3、AWS Glue API 許可，以及存取 Lake Formation 的許可。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ScriptAccess",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::*.amzn-s3-demo-bucket/scripts"
      ]
    }
  ]
}
```

```
    ]
  },
  {
    "Sid": "LoggingAccess",
    "Effect": "Allow",
    "Action": [
      "s3:PutObject"
    ],
    "Resource": [
      "arn:aws:s3:::amzn-s3-demo-bucket/logs/*"
    ]
  },
  {
    "Sid": "GlueCatalogAccess",
    "Effect": "Allow",
    "Action": [
      "glue:Get*",
      "glue:Create*",
      "glue:Update*"
    ],
    "Resource": [
      "*"
    ]
  },
  {
    "Sid": "LakeFormationAccess",
    "Effect": "Allow",
    "Action": [
      "lakeformation:GetDataAccess"
    ],
    "Resource": [
      "*"
    ]
  }
]
```

步驟 2.1 設定 Lake Formation 許可

- 從 S3 讀取資料的 Spark 任務需要 Lake Formation SELECT 許可。
- 在 S3 中寫入/刪除資料的 Spark 任務需要 Lake Formation ALL (SUPER) 許可。
- 與 Glue Data Catalog 互動的 Spark 任務需要適當的 DESCRIBE、ALTER、DROP 許可。

如需詳細資訊，請參閱[授予 Data Catalog 資源的許可](#)。

步驟 3：使用 Lake Formation 初始化完整資料表存取的 Spark 工作階段

先決條件

AWS Glue Data Catalog 必須設定為中繼存放區，才能存取 Lake Formation 資料表。

設定下列設定，將 Glue 目錄設定為中繼存放區：

```
--conf spark.sql.catalogImplementation=hive
--conf
  spark.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveMetastore
```

如需為 EMR Serverless 啟用 Data Catalog 的詳細資訊，請參閱 [EMR Serverless 的中繼存放區組態](#)。

若要存取向 AWS Lake Formation 註冊的資料表，需要在 Spark 初始化期間設定下列組態，以將 Spark 設定為使用 AWS Lake Formation 憑證。

Hive

```
--conf
  spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.glue.accesscontrol.AWSLakeFormationS3CredentialsResolver
--conf spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true
--conf spark.hadoop.fs.s3.folderObject.autoAction.disabled=true
--conf spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true
--conf spark.sql.catalog.createDirectoryAfterTable.enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
```

Iceberg

```
--conf spark.sql.catalog.spark_catalog=org.apache.iceberg.spark.SparkSessionCatalog
--conf spark.sql.catalog.spark_catalog.warehouse=S3_DATA_LOCATION
--conf spark.sql.catalog.spark_catalog.client.region=REGION
--conf spark.sql.catalog.spark_catalog.type=glue
--conf spark.sql.catalog.spark_catalog.glue.account-id=ACCOUNT_ID
--conf spark.sql.catalog.spark_catalog.glue.lakeformation-enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
```

Delta Lake

```
--conf
  spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.glue.accesscontrol.AWSLakeFormationS3CredentialsResolver
```

```
--conf spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true
--conf spark.hadoop.fs.s3.folderObject.autoAction.disabled=true
--conf spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true
--conf spark.sql.catalog.createDirectoryAfterTable.enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
```

Hudi

```
--conf
  spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.glue.accesscontrol.AWSLakeFormationAccessControl
--conf spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true
--conf spark.hadoop.fs.s3.folderObject.autoAction.disabled=true
--conf spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true
--conf spark.sql.catalog.createDirectoryAfterTable.enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
--conf spark.jars=/usr/lib/hudi/hudi-spark-bundle.jar
--conf spark.sql.extensions=org.apache.spark.sql.hudi.HoodieSparkSessionExtension
--conf
  spark.sql.catalog.spark_catalog=org.apache.spark.sql.hudi.catalog.HoodieCatalog
--conf spark.serializer=org.apache.spark.serializer.KryoSerializer
```

- `spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.glue.accesscontrol.AWSLakeFormationAccessControl`：設定 EMR Filesystem (EMRFS) 或 EMR S3A 以使用 AWS Lake Formation 註冊資料表的 Lake Formation S3 登入資料。如果未註冊資料表，請使用任務的執行時期角色憑證。
- `spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true` 和 `spark.hadoop.fs.s3.folderObject.autoAction.disabled=true`：將 EMRFS 設定為在建立 S3 資料夾時使用內容類型標頭 `application/x-directory`，而非 `$folder$` 尾碼。這是讀取 Lake Formation 資料表時的必要項目，因為 Lake Formation 憑證不允許讀取具有 `$folder$` 尾碼的資料表資料夾。
- `spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true`：將 Spark 設定為在建立之前略過驗證資料表位置的空白。這對於 Lake Formation 註冊的資料表是必要的，因為 Lake Formation 登入資料只有在 Glue Data Catalog 資料表建立後才能使用。如果沒有此組態，任務的執行時期角色憑證將驗證空白資料表位置。
- `spark.sql.catalog.createDirectoryAfterTable.enabled=true`：將 Spark 設定為在 Hive 中繼存放區中建立資料表之後建立 Amazon S3 資料夾。Lake Formation 註冊的資料表需要此項目，因為建立 S3 資料夾的 Lake Formation 登入資料只有在 Glue Data Catalog 資料表建立後才能使用。

- `spark.sql.catalog.dropDirectoryBeforeTable.enabled=true` : 設定 Spark 在 Hive 中繼存放區中刪除資料表之前捨棄 S3 資料夾。這對於 Lake Formation 註冊的資料表是必要的，因為從 Glue Data Catalog 刪除資料表後，無法使用 Lake Formation 登入資料來捨棄 S3 資料夾。
- `spark.sql.catalog.<catalog>.glue.lakeformation-enabled=true` : 設定 Iceberg 目錄以使用 AWS Lake Formation 註冊資料表的 Lake Formation S3 登入資料。如果未註冊資料表，請使用預設環境憑證。

在 SageMaker Unified Studio 中設定完整資料表存取模式

若要從 JupyterLab 筆記本中的互動式 Spark 工作階段存取 Lake Formation 註冊的資料表，請使用相容性許可模式。使用 `%%configure` 魔術命令來設定 Spark 組態。根據資料表類型選擇組態：

For Hive tables

```
%%configure -f
{
  "conf": {
    "spark.hadoop.fs.s3.credentialsResolverClass":
"com.amazonaws.glue.accesscontrol.AWSLakeFormationCredentialResolver",
    "spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject": true,
    "spark.hadoop.fs.s3.folderObject.autoAction.disabled": true,
    "spark.sql.catalog.skipLocationValidationOnCreateTable.enabled": true,
    "spark.sql.catalog.createDirectoryAfterTable.enabled": true,
    "spark.sql.catalog.dropDirectoryBeforeTable.enabled": true
  }
}
```

For Iceberg tables

```
%%configure -f
{
  "conf": {
    "spark.sql.catalog.spark_catalog":
"org.apache.iceberg.spark.SparkSessionCatalog",
    "spark.sql.catalog.spark_catalog.warehouse": "S3_DATA_LOCATION",
    "spark.sql.catalog.spark_catalog.client.region": "REGION",
    "spark.sql.catalog.spark_catalog.type": "glue",
    "spark.sql.catalog.spark_catalog.glue.account-id": "ACCOUNT_ID",
    "spark.sql.catalog.spark_catalog.glue.lakeformation-enabled": "true",
    "spark.sql.catalog.dropDirectoryBeforeTable.enabled": "true"
  }
}
```

```
}

```

For Delta Lake tables

```
%%configure -f
{
  "conf": {
    "spark.hadoop.fs.s3.credentialsResolverClass":
"com.amazonaws.glue.accesscontrol.AWSLakeFormationCredentialResolver",
    "spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject": true,
    "spark.hadoop.fs.s3.folderObject.autoAction.disabled": true,
    "spark.sql.catalog.skipLocationValidationOnCreateTable.enabled": true,
    "spark.sql.catalog.createDirectoryAfterTable.enabled": true,
    "spark.sql.catalog.dropDirectoryBeforeTable.enabled": true
  }
}
```

For Hudi tables

```
%%configure -f
{
  "conf": {
    "spark.hadoop.fs.s3.credentialsResolverClass":
"com.amazonaws.glue.accesscontrol.AWSLakeFormationCredentialResolver",
    "spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject": true,
    "spark.hadoop.fs.s3.folderObject.autoAction.disabled": true,
    "spark.sql.catalog.skipLocationValidationOnCreateTable.enabled": true,
    "spark.sql.catalog.createDirectoryAfterTable.enabled": true,
    "spark.sql.catalog.dropDirectoryBeforeTable.enabled": true,
    "spark.jars": "/usr/lib/hudi/hudi-spark-bundle.jar",
    "spark.sql.extensions":
"org.apache.spark.sql.hudi.HoodieSparkSessionExtension",
    "spark.sql.catalog.spark_catalog":
"org.apache.spark.sql.hudi.catalog.HoodieCatalog",
    "spark.serializer": "org.apache.spark.serializer.KryoSerializer"
  }
}
```

取代預留位置：

- `S3_DATA_LOCATION`：您的 S3 儲存貯體路徑

- REGION : AWS region (例如 us-east-1)
- ACCOUNT_ID : AWS 您的帳戶 ID

Note

必須先設定這些組態，才能在筆記本中執行任何 Spark 操作。

受支援的 操作

這些操作將使用 AWS Lake Formation 登入資料來存取資料表資料。

- CREATE TABLE
- ALTER TABLE
- INSERT INTO
- INSERT OVERWRITE
- UPDATE
- 合併為
- DELETE FROM
- ANALYZE TABLE
- REPAIR TABLE
- DROP TABLE
- Spark 資料來源查詢
- Spark 資料來源寫入

Note

上面未列出的操作將繼續使用 IAM 許可來存取資料表資料。

考量事項

- 如果 Hive 資料表是使用未啟用完整資料表存取的任務建立，而且未插入任何記錄，則具有完整資料表存取的任務後續讀取或寫入將會失敗。這是因為沒有完整資料表存取權的 EMR Spark \$folder\$ 會將尾碼新增至資料表資料夾名稱。若要解決此問題，您可以：

- 從未啟用 FTA 的任務中將至少一列插入資料表。
- 將未啟用 FTA 的任務設定為 S3 中資料夾名稱不使用 `$folder$` 尾碼。這可以透過設定 Spark 組態 `spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true` 來實現。
- `s3://path/to/table/table_name` 使用 S3 主控台或 AWS S3 CLI 在資料表位置建立 AWS S3 資料夾。
- 從 Amazon EMR 7.8.0 版開始的 EMR 檔案系統 (EMRFS) 和從 Amazon EMR 7.10.0 版開始的 S3A 檔案系統支援完整資料表存取。
- Hive、Iceberg、Delta 和 Hudi 資料表支援完整資料表存取。
- Hudi FTA Write Support 考量事項：
 - Hudi FTA 寫入要求在任務執行期間使用 `HoodieCredentialedHadoopStorage` 進行登入資料販賣。在執行 Hudi 任務時設定下列組態：
`hoodie.storage.class=org.apache.spark.sql.hudi.storage.HoodieCredentialedHadoopStorage`
 - Hudi 的完整資料表存取 (FTA) 寫入支援從 Amazon EMR 7.12 版開始提供。
 - Hudi FTA 寫入支援目前僅適用於預設 Hudi 組態。自訂或非預設 Hudi 設定可能未完全支援，並可能導致非預期的行為。
 - 在 FTA 寫入模式下，目前不支援叢集 Hudi Merge-On-Read (MOR) 資料表。
- 參考具有 Lake Formation 精細存取控制 (FGAC) 規則或 Glue Data Catalog 檢視的資料表的任務將會失敗。若要查詢具有 FGAC 規則或 Glue Data Catalog View 的資料表，您必須使用 FGAC 模式。您可以按照 AWS 文件中概述的步驟啟用 FGAC 模式：[使用 EMR Serverless 搭配 AWS Lake Formation 進行精細存取控制](#)。
- 完整資料表存取不支援 Spark 串流。
- 將 Spark DataFrame 寫入 Lake Formation 資料表時，Hive 和 Iceberg 資料表僅支援 APPEND 模式：`df.write.mode("append").saveAsTable(table_name)`
- 建立外部資料表需要 IAM 許可。
- 由於 Lake Formation 會暫時快取 Spark 任務中的登入資料，因此目前正在執行的 Spark 批次任務或互動式工作階段可能不會反映許可變更。
- 您必須使用使用者定義的角色，而不是角色的服務連結角色：[Lake Formation 要求](#)。

Hudi FTA 寫入支援 - 支援的操作

下表顯示完整資料表存取模式下 Hudi Copy-On-Write(COW) 和 Merge-On-Read(MOR) 資料表支援的寫入操作：

Hudi FTA 支援的寫入操作

資料表類型	作業	SQL 寫入命令	狀態
COW	INSERT	INSERT INTO TABLE	支援
COW	INSERT	INSERT INTO TABLE - PARTITION (靜態、動態)	支援
COW	INSERT	INSERT OVERWRITE	支援
COW	INSERT	INSERT OVERWRITE - PARTITION (靜態、動態)	支援
UPDATE	UPDATE	UPDATE TABLE	支援
COW	UPDATE	更新資料表 - 變更分割區	不支援
DELETE	DELETE	DELETE FROM TABLE	支援
ALTER	ALTER	ALTER TABLE - 重新命名為	不支援
COW	ALTER	ALTER TABLE - 設定 TBLPROPERTIES	支援
COW	ALTER	ALTER TABLE - 取消設定 TBLPROPERTIES	支援

資料表類型	作業	SQL 寫入命令	狀態
COW	ALTER	ALTER 資料表 - ALTER 資料欄	支援
COW	ALTER	ALTER TABLE - 新增資料欄	支援
COW	ALTER	ALTER TABLE - 新增分割區	支援
COW	ALTER	ALTER TABLE - 捨棄分割區	支援
COW	ALTER	ALTER TABLE - 復原分割區	支援
COW	ALTER	修復資料表同步 分割區	支援
DROP	DROP	DROP TABLE	支援
COW	DROP	DROP TABLE - PURGE	支援
CREATE	CREATE	CREATE TABLE - 受管	支援
COW	CREATE	建立資料表 - 分 割區依據	支援
COW	CREATE	如果不存在，則 建立資料表	支援
COW	CREATE	CREATE TABLE LIKE	支援
COW	CREATE	CREATE TABLE AS SELECT	支援

資料表類型	作業	SQL 寫入命令	狀態
CREATE	CREATE	使用 LOCATION 建立資料表 - 外部資料表	不支援
DATAFRAME (INSERT)	DATAFRAME(INSERT)	saveAsTable. Overwrite	支援
COW	DATAFRAME(INSERT)	saveAsTable. Append	不支援
COW	DATAFRAME(INSERT)	saveAsTable. Ignore	支援
COW	DATAFRAME(INSERT)	saveAsTable. ErrorIfExists	支援
COW	DATAFRAME(INSERT)	saveAsTable - 外部資料表 (路徑)	不支援
COW	DATAFRAME(INSERT)	save(path) - DF v1	不支援
MOR	INSERT	INSERT INTO TABLE	支援
MOR	INSERT	INSERT IN TO TABLE - PARTITION (靜 態、動態)	支援
MOR	INSERT	INSERT OVERWRITE	支援

資料表類型	作業	SQL 寫入命令	狀態
MOR	INSERT	INSERT OVERWRITE - PARTITION (靜 態、動態)	支援
UPDATE	UPDATE	UPDATE TABLE	支援
MOR	UPDATE	更新資料表 - 變 更分割區	不支援
DELETE	DELETE	DELETE FROM TABLE	支援
ALTER	ALTER	ALTER TABLE - 重新命名為	不支援
MOR	ALTER	ALTER TABLE - 設定 TBLPROPER TIES	支援
MOR	ALTER	ALTER TABLE - 取消設定 TBLPROPER TIES	支援
MOR	ALTER	ALTER 資料表 - ALTER 資料欄	支援
MOR	ALTER	ALTER TABLE - 新增資料欄	支援
MOR	ALTER	ALTER TABLE - 新增分割區	支援
MOR	ALTER	ALTER TABLE - 捨棄分割區	支援

資料表類型	作業	SQL 寫入命令	狀態
MOR	ALTER	ALTER TABLE - 復原分割區	支援
MOR	ALTER	修復資料表同步分割區	支援
DROP	DROP	DROP TABLE	支援
MOR	DROP	DROP TABLE - PURGE	支援
CREATE	CREATE	CREATE TABLE - 受管	支援
MOR	CREATE	建立資料表 - 分割區依據	支援
MOR	CREATE	如果不存在，則建立資料表	支援
MOR	CREATE	CREATE TABLE LIKE	支援
MOR	CREATE	CREATE TABLE AS SELECT	支援
CREATE	CREATE	使用 LOCATION 建立資料表 - 外部資料表	不支援
DATAFRAME (UPSERT)	DATAFRAME(UPSERT)	saveAsTable.Overwrite	支援
MOR	DATAFRAME(UPSERT)	saveAsTable.Append	不支援
MOR	DATAFRAME(UPSERT)	saveAsTable.Ignore	支援

資料表類型	作業	SQL 寫入命令	狀態
MOR	DATAFRAME(UPSERT)	saveAsTable.ErrorIfExists	支援
MOR	DATAFRAME(UPSERT)	saveAsTable - 外部資料表 (路徑)	不支援
MOR	DATAFRAME(UPSERT)	save(path) - DF v1	不支援
DATAFRAME (刪除)	DATAFRAME (刪除)	saveAsTable.Append	不支援
MOR	DATAFRAME (刪除)	saveAsTable - 外部資料表 (路徑)	不支援
MOR	DATAFRAME (刪除)	save(path) - DF v1	不支援
DATAFRAME (BULK_INSERT)	DATAFRAME(BULK_INSERT)	saveAsTable.Overwrite	支援
MOR	DATAFRAME(BULK_INSERT)	saveAsTable.Append	不支援
MOR	DATAFRAME(BULK_INSERT)	saveAsTable.Ignore	支援
MOR	DATAFRAME(BULK_INSERT)	saveAsTable.ErrorIfExists	支援
MOR	DATAFRAME(BULK_INSERT)	saveAsTable - 外部資料表 (路徑)	不支援

資料表類型	作業	SQL 寫入命令	狀態
MOR	DATAFRAME(BULK_INSERT)	save(path) - DF v1	不支援

使用 EMR Serverless 搭配 AWS Lake Formation 進行精細存取控制

概觀

使用 Amazon EMR 7.2.0 版及更高版本，利用 AWS Lake Formation 對 S3 支援的 Data Catalog 資料表套用精細存取控制。此功能可讓您設定 Amazon EMR Serverless Spark 任務中 read 查詢的資料表、資料列、資料欄和儲存格層級存取控制。若要設定 Apache Spark 批次任務和互動式工作階段的精細存取控制，請使用 EMR Studio。請參閱下列各節，進一步了解 Lake Formation 以及如何搭配 EMR Serverless 使用。

搭配使用 Amazon EMR Serverless AWS Lake Formation 會產生額外費用。如需詳細資訊，請參閱 [Amazon EMR 定價](#)。

EMR Serverless 如何使用 AWS Lake Formation

搭配 Lake Formation 使用 EMR Serverless 可讓您對每個 Spark 任務強制執行一層許可，以在 EMR Serverless 執行任務時套用 Lake Formation 許可控制。EMR Serverless 使用 [Spark 資源描述](#) 檔來建立兩個描述檔，以有效地執行任務。使用者設定檔會執行使用者提供的程式碼，而系統設定檔則會強制執行 Lake Formation 政策。如需詳細資訊，請參閱 [什麼是 AWS Lake Formation](#) 以及 [考量和限制](#)。

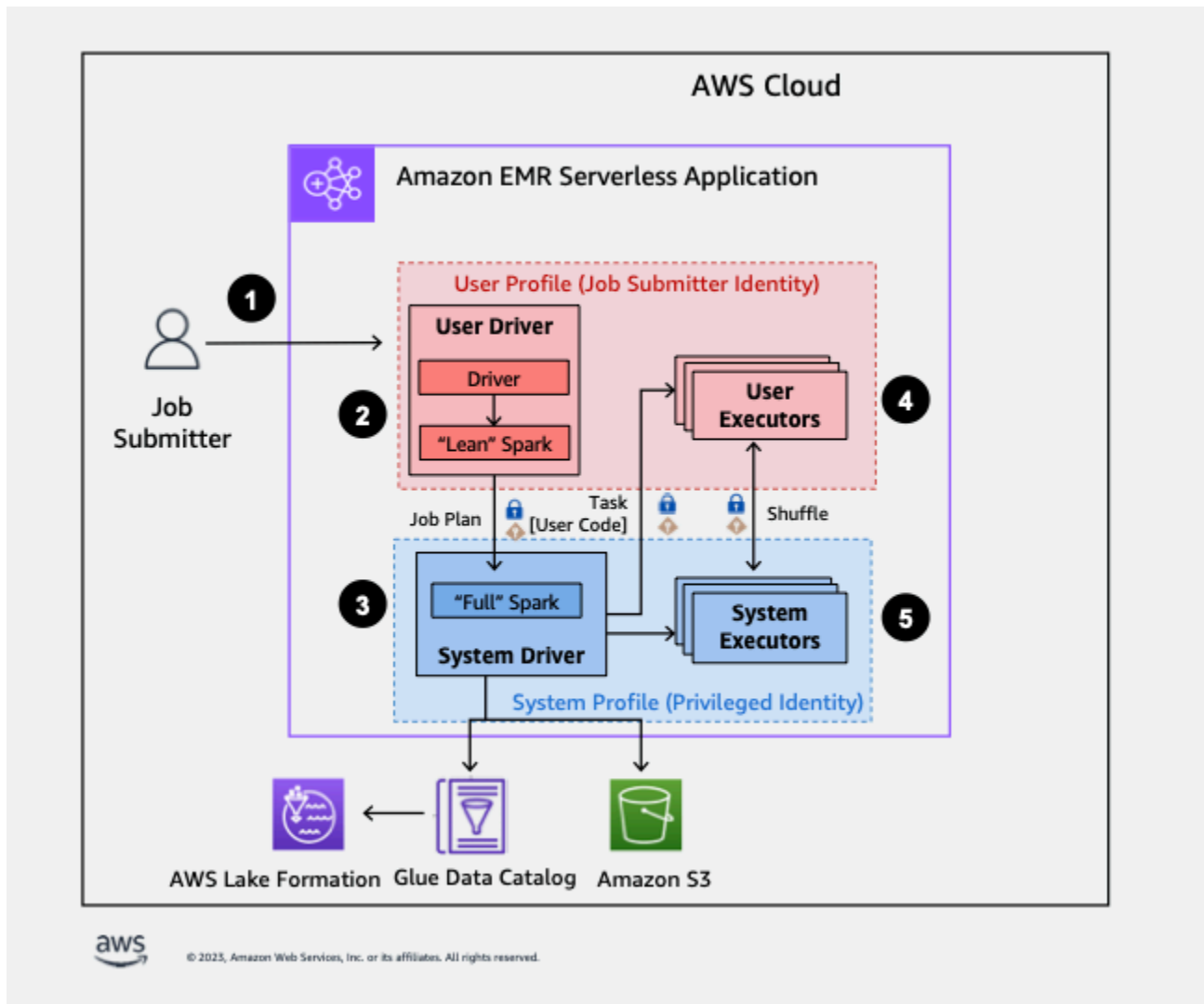
當您搭配 Lake Formation 使用預先初始化的容量時，我們建議您至少有兩個 Spark 驅動程式。每個啟用 Lake Formation 的任務都會使用兩個 Spark 驅動程式，一個用於使用者設定檔，另一個用於系統設定檔。為了獲得最佳效能，如果您不使用 Lake Formation，請使用已啟用 Lake Formation 任務的驅動程式數量的兩倍。

當您在 EMR Serverless 上執行 Spark 任務時，也請考慮動態配置對資源管理和叢集效能的影響。每個資源設定檔的執行器數目 `spark.dynamicAllocation.maxExecutors` 上限組態適用於使用者和系統執行器。如果您將該數目設定為等於允許的執行器數量上限，您的任務執行可能會因為使用所有可用資源的一種執行器類型而停滯，這會在您執行任務時防止其他執行器。

因此，您不會耗盡資源，EMR Serverless 會將每個資源設定檔的預設執行器數目上限設定為 `spark.dynamicAllocation.maxExecutors` 值的 90%。當您 `spark.dynamicAllocation.maxExecutorsRatio` 以 0 到 1 之間的值指定時，您可以覆寫此組態。此外，也請設定下列屬性，以最佳化資源配置和整體效能：

- `spark.dynamicAllocation.cachedExecutorIdleTimeout`
- `spark.dynamicAllocation.shuffleTracking.timeout`
- `spark.cleaner.periodicGC.interval`

以下是 EMR Serverless 如何存取 Lake Formation 安全政策所保護資料的高階概觀。



1. 使用者將 Spark 任務提交至 AWS Lake Formation 已啟用 EMR Serverless 應用程式。
2. EMR Serverless 會將任務傳送給使用者驅動程式，並在使用者設定檔中執行任務。使用者驅動程式會執行 Spark 的精簡版本，該版本無法啟動任務、請求執行器、存取 S3 或 Glue Catalog。其會建置任務計畫。
3. EMR Serverless 會設定第二個稱為系統驅動程式的驅動程式，並在系統設定檔中執行它（具有特殊權限身分）。EMR Serverless 會在兩個驅動程式之間設定加密的 TLS 頻道以進行通訊。使用者驅動程式使用該頻道將任務計畫傳送至系統驅動程式。系統驅動程式不會執行使用者提交的程式。

碼。其會執行完整的 Spark，並與 S3 和 Data Catalog 通訊以進行資料存取。其會請求執行器，並將任務計畫編譯成一系列的執行階段。

- 然後，EMR Serverless 會使用使用者驅動程式或系統驅動程式在執行器上執行階段。任何階段的使用者程式碼只會在使用者設定檔執行器上執行。
- 從受保護的資料目錄資料表 AWS Lake Formation 或套用安全篩選條件的資料表讀取資料的階段，會委派給系統執行器。

在 Amazon EMR 中啟用 Lake Formation

若要啟用 Lake Formation，請在[建立 EMR Serverless 應用程式](#)時，將 Runtime-configuration 參數的 spark-defaults 分類 spark.emr-serverless.lakeformation.enabled 設為 true。

```
aws emr-serverless create-application \  
  --release-label emr-7.13.0 \  
  --runtime-configuration '{  
    "classification": "spark-defaults",  
    "properties": {  
      "spark.emr-serverless.lakeformation.enabled": "true"  
    }  
  }' \  
  --type "SPARK"
```

您也可以可以在 EMR Studio 中建立新應用程式時啟用 Lake Formation。選擇使用 Lake Formation 進行精細存取控制，可在其他組態下使用。

使用 Lake Formation 搭配 EMR Serverless 時，預設會啟用[工作者間加密](#)，因此您不需要再次明確啟用工作者間加密。

為 Spark 任務啟用 Lake Formation

若要為個別 Spark 任務啟用 Lake Formation，請在使用時 spark.emr-serverless.lakeformation.enabled 將設定為 truespark-submit。

```
--conf spark.emr-serverless.lakeformation.enabled=true
```

任務執行時期角色 IAM 許可

Lake Formation 許可控制對 AWS Glue Data Catalog 資源、Amazon S3 位置和這些位置基礎資料的存取。IAM 許可可控制對 Lake Formation 和 AWS Glue API 和資源的存取。雖然您可能具有 Lake

Formation 許可來存取 Data Catalog (SELECT) 中的資料表，但是如果您沒有 `glue:Get*` API 操作的 IAM 許可，您的操作會失敗。

以下範例政策說明如何提供 IAM 許可來存取 S3 中的指令碼 (將日誌上傳至 S3)、AWS Glue API 許可以及用於存取 Lake Formation 的許可。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ScriptAccess",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3::*.amzn-s3-demo-bucket/scripts",
        "arn:aws:s3::*.amzn-s3-demo-bucket/*"
      ]
    },
    {
      "Sid": "LoggingAccess",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/logs/*"
      ]
    },
    {
      "Sid": "GlueCatalogAccess",
      "Effect": "Allow",
      "Action": [
        "glue:Get*",
        "glue:Create*",
        "glue:Update*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

```
    ]
  },
  {
    "Sid": "LakeFormationAccess",
    "Effect": "Allow",
    "Action": [
      "lakeformation:GetDataAccess"
    ],
    "Resource": [
      "*"
    ]
  }
]
```

設定任務執行時期角色的 Lake Formation 許可

首先，向 Lake Formation 註冊 Hive 資料表的位置。然後在所需的資料表上建立任務執行時期角色的許可。如需 Lake Formation 的詳細資訊，請參閱[什麼是 AWS Lake Formation?](#) 《AWS Lake Formation 開發人員指南》中的。

設定 Lake Formation 許可後，請在 Amazon EMR Serverless 上提交 Spark 任務。如需 Spark 任務的詳細資訊，請參閱[Spark 範例](#)。

提交任務執行

完成 Lake Formation 授予的設定後，您可以在[EMR Serverless 上提交 Spark 任務](#)。以下章節顯示如何設定和提交任務執行屬性的範例。

許可要求

資料表未在 中註冊 AWS Lake Formation

對於未向 註冊的資料表 AWS Lake Formation，任務執行期角色會存取 AWS Glue Data Catalog 和 Amazon S3 中的基礎資料表資料。這需要任務執行期角色具有 Glue 和 Amazon S3 AWS 操作的適當 IAM 許可。

在 中註冊的資料表 AWS Lake Formation

對於向 註冊的資料表 AWS Lake Formation，任務執行時間角色會存取 AWS Glue Data Catalog 中繼資料，而 Lake Formation 提供的臨時憑證則會存取 Amazon S3 中的基礎資料表資料。執行 操作所需

的 Lake Formation 許可取決於 Spark 任務啟動的 AWS Glue Data Catalog 和 Amazon S3 API 呼叫，可以摘要如下：

- DESCRIBE 許可允許執行期角色讀取 Data Catalog 中的資料表或資料庫中繼資料
- ALTER 許可允許執行期角色修改 Data Catalog 中的資料表或資料庫中繼資料
- DROP 許可允許執行期角色從 Data Catalog 刪除資料表或資料庫中繼資料
- SELECT 許可允許執行期角色從 Amazon S3 讀取資料表資料
- INSERT 許可允許執行期角色將資料表資料寫入 Amazon S3
- DELETE 許可允許執行期角色從 Amazon S3 刪除資料表資料

Note

當 Spark 任務呼叫 AWS Glue 擷取資料表中繼資料和 Amazon S3 擷取資料表資料時，Lake Formation 會延遲評估許可。在 Spark 進行需要缺少許可的 AWS Glue 或 Amazon S3 呼叫之前，使用執行時間角色且許可不足的任務不會失敗。

Note

在下列支援的資料表矩陣中：

- 標示為 Supported 的操作只會使用 Lake Formation 登入資料來存取向 Lake Formation 註冊之資料表的資料表資料。如果 Lake Formation 許可不足，操作將不會回復為執行時間角色登入資料。對於未向 Lake Formation 註冊的資料表，任務執行期角色登入資料會存取資料表資料。
- 在 Amazon S3 位置上標記為支援 IAM 許可的操作不會使用 Lake Formation 登入資料來存取 Amazon S3 中的基礎資料表資料。若要執行這些操作，任務執行時間角色必須具有存取資料表資料所需的 Amazon S3 IAM 許可，無論資料表是否已向 Lake Formation 註冊。

Hive

作業	AWS Lake Formation 許可	支援狀態
SELECT	SELECT	支援
CREATE TABLE	CREATE_TABLE	支援
CREATE TABLE LIKE	CREATE_TABLE	在 Amazon S3 位置上支援 IAM 許可
CREATE TABLE AS SELECT	CREATE_TABLE	在 Amazon S3 位置上支援 IAM 許可
DESCRIBE TABLE	DESCRIBE	支援
SHOW TBLPROPERTIES	DESCRIBE	支援
SHOW COLUMNS	DESCRIBE	支援
SHOW PARTITIONS	DESCRIBE	支援
SHOW CREATE TABLE	DESCRIBE	支援
修改資料表 tablename	SELECT 和 ALTER	支援
更改資料表tablename 集位置	-	不支援
更改資料表tablename 新增分割區	SELECT、INSERT 和 ALTER	支援
REPAIR TABLE	SELECT 和 ALTER	支援
載入資料		不支援
INSERT	INSERT 和 ALTER	支援

作業	AWS Lake Formation 許可	支援狀態
INSERT OVERWRITE	SELECT、INSERT、DELETE 和 ALTER	支援
DROP TABLE	SELECT、DROP、DELETE 和 ALTER	支援
TRUNCATE TABLE	SELECT、INSERT、DELETE 和 ALTER	支援
Dataframe Writer V1	與對應的 SQL 操作相同	將資料附加至現有資料表時支援。如需詳細資訊，請參閱 考量事項和限制
Dataframe Writer V2	與對應的 SQL 操作相同	將資料附加至現有資料表時支援。如需詳細資訊，請參閱 考量事項和限制

Iceberg

作業	AWS Lake Formation 許可	支援狀態
SELECT	SELECT	支援
CREATE TABLE	CREATE_TABLE	支援
CREATE TABLE LIKE	CREATE_TABLE	在 Amazon S3 位置上支援 IAM 許可
CREATE TABLE AS SELECT	CREATE_TABLE	在 Amazon S3 位置上支援 IAM 許可
將資料表取代為選取	SELECT、INSERT 和 ALTER	支援
DESCRIBE TABLE	DESCRIBE	在 Amazon S3 位置上支援 IAM 許可

作業	AWS Lake Formation 許可	支援狀態
SHOW TBLPROPER TIES	DESCRIBE	在 Amazon S3 位置上支援 IAM 許可
SHOW CREATE TABLE	DESCRIBE	在 Amazon S3 位置上支援 IAM 許可
ALTER TABLE	SELECT、INSERT 和 ALTER	支援
ALTER TABLE SET LOCATION	SELECT、INSERT 和 ALTER	在 Amazon S3 位置上支援 IAM 許可
更改 排序的資料表寫入	SELECT、INSERT 和 ALTER	在 Amazon S3 位置上支援 IAM 許可
更改 分佈的資料表寫入	SELECT、INSERT 和 ALTER	在 Amazon S3 位置上支援 IAM 許可
更改資料表重新命名資料表	CREATE_TABLE 和 DROP	支援
INSERT INTO	SELECT、INSERT 和 ALTER	支援
INSERT OVERWRITE	SELECT、INSERT 和 ALTER	支援
DELETE	SELECT、INSERT 和 ALTER	支援
UPDATE	SELECT、INSERT 和 ALTER	支援
合併為	SELECT、INSERT 和 ALTER	支援
DROP TABLE	SELECT、DELETE 和 DROP	支援

作業	AWS Lake Formation 許可	支援狀態
DataFrame 寫入器 V1	-	不支援
DataFrame 寫入器 V2	與對應的 SQL 操作相同	將資料附加至現有資料表時支援。 如需詳細資訊，請參閱 考量事項和限制 。
中繼資料表	SELECT	支援。某些資料表會隱藏。如需詳細資訊，請參閱 考量事項和限制 。
預存程序	-	支援符合下列條件的資料表： <ul style="list-style-type: none"> 資料表未在 中註冊 AWS Lake Formation 不使用 register_table 和 的資料表 migrate <p>如需詳細資訊，請參閱考量事項和限制。</p>

Iceberg 的 Spark 組態：下列範例示範如何使用 Iceberg 設定 Spark。若要執行 Iceberg 任務，請提供下列 spark-submit 屬性。

```
--conf spark.sql.catalog.spark_catalog=org.apache.iceberg.spark.SparkSessionCatalog
--conf spark.sql.catalog.spark_catalog.warehouse=<S3_DATA_LOCATION>
--conf spark.sql.catalog.spark_catalog.glue.account-id=<ACCOUNT_ID>
--conf spark.sql.catalog.spark_catalog.client.region=<REGION>
--conf spark.sql.catalog.spark_catalog.glue.endpoint=https://
glue.<REGION>.amazonaws.com
```

Hudi

作業	AWS Lake Formation 許可	支援狀態
SELECT	SELECT	支援

作業	AWS Lake Formation 許可	支援狀態
CREATE TABLE	CREATE_TABLE	在 Amazon S3 位置上支援 IAM 許可
CREATE TABLE LIKE	CREATE_TABLE	在 Amazon S3 位置上支援 IAM 許可
CREATE TABLE AS SELECT	-	不支援
DESCRIBE TABLE	DESCRIBE	在 Amazon S3 位置上支援 IAM 許可
SHOW TBLPROPERTIES	DESCRIBE	在 Amazon S3 位置上支援 IAM 許可
SHOW COLUMNS	DESCRIBE	在 Amazon S3 位置上支援 IAM 許可
SHOW CREATE TABLE	DESCRIBE	在 Amazon S3 位置上支援 IAM 許可
ALTER TABLE	SELECT	在 Amazon S3 位置上支援 IAM 許可
INSERT INTO	SELECT 和 ALTER	在 Amazon S3 位置上支援 IAM 許可
INSERT OVERWRITE	SELECT 和 ALTER	在 Amazon S3 位置上支援 IAM 許可
DELETE	-	不支援
UPDATE	-	不支援
合併為	-	不支援
DROP TABLE	SELECT 和 DROP	在 Amazon S3 位置上支援 IAM 許可

作業	AWS Lake Formation 許可	支援狀態
DataFrame 寫入器 V1	-	不支援
DataFrame 寫入器 V2	與對應的 SQL 操作相同	在 Amazon S3 位置上支援 IAM 許可
中繼資料表	-	不支援
資料表維護和公用程式功能	-	不支援

下列範例使用 Hudi 設定 Spark，指定檔案位置和使用所需的其他屬性。

Hudi 的 Spark 組態：此程式碼片段用於筆記本時，會指定 Hudi Spark 套件 JAR 檔案的路徑，以啟用 Spark 中的 Hudi 功能。它也會將 Spark 設定為使用 AWS Glue Data Catalog 作為中繼存放區。

```
%%configure -f
{
  "conf": {
    "spark.jars": "/usr/lib/hudi/hudi-spark-bundle.jar",
    "spark.hadoop.hive.metastore.client.factory.class":
"com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory",
    "spark.serializer": "org.apache.spark.serializer.JavaSerializer",
    "spark.sql.catalog.spark_catalog":
"org.apache.spark.sql.hudi.catalog.HoodieCatalog",
    "spark.sql.extensions":
"org.apache.spark.sql.hudi.HoodieSparkSessionExtension"
  }
}
```

Hudi 與 Glue AWS 的 Spark 組態：此程式碼片段用於筆記本時，可讓 Hudi 成為支援的資料湖格式，並確保 Hudi 程式庫和相依性可用。

```
%%configure
{
  "--conf": "spark.serializer=org.apache.spark.serializer.JavaSerializer --conf
spark.sql.catalog.spark_catalog=org.apache.spark.sql.hudi.catalog.HoodieCatalog --
conf
spark.sql.extensions=org.apache.spark.sql.hudi.HoodieSparkSessionExtension",
  "--datalake-formats": "hudi",
  "--enable-glue-datacatalog": True,
```

```

    "--enable-lakeformation-fine-grained-access": "true"
  }

```

Delta Lake

作業	AWS Lake Formation 許可	支援狀態
SELECT	SELECT	支援
CREATE TABLE	CREATE_TABLE	支援
CREATE TABLE LIKE	-	不支援
CREATE TABLE AS SELECT	CREATE_TABLE	支援
將資料表取代為選取	SELECT、INSERT 和 ALTER	支援
DESCRIBE TABLE	DESCRIBE	在 Amazon S3 位置上支援 IAM 許可
SHOW TBLPROPERTIES	DESCRIBE	在 Amazon S3 位置上支援 IAM 許可
SHOW COLUMNS	DESCRIBE	在 Amazon S3 位置上支援 IAM 許可
SHOW CREATE TABLE	DESCRIBE	在 Amazon S3 位置上支援 IAM 許可
ALTER TABLE	SELECT 和 INSERT	支援
ALTER TABLE SET LOCATION	SELECT 和 INSERT	在 Amazon S3 位置上支援 IAM 許可
修改資料表tablename 叢集	SELECT 和 INSERT	在 Amazon S3 位置上支援 IAM 許可

偵錯任務

Note

使用此功能時，存取可能包含敏感、未篩選資訊之系統設定檔工作者的 stdout 和 stderr 日誌。下列許可應僅用於存取非生產資料。對於為搭配生產任務使用而建立的應用程式，我們強烈建議您僅將這些許可新增至具有更高資料存取的管理員或使用者。

使用 EMR-7.3.0 和更新版本，EMR Serverless 為已啟用 Lake Formation 的批次任務啟用自我偵錯功能。若要這樣做，請使用 [GetDashboardForJobRun](#) API 中的新參數 `accessSystemProfileLogs`。如果 `accessSystemProfileLogs` 設為 `true`，您可以存取系統設定檔工作者的 stdout 和 stderr 日誌，可用於偵錯已啟用 Lake Formation 的 EMR Serverless 批次任務。

```
aws emr-serverless get-dashboard-for-job-run \
  --application-id application-id
  --job-run-id job-run-id
  --access-system-profile-logs
```

所需的許可

想要使用 `GetDashboardForJobRun` 偵錯已啟用 Lake Formation 的批次任務的委託人必須具有下列額外許可：

```
{
  "Sid": "AccessSystemProfileLogs",
  "Effect": "Allow",
  "Action": [
    "emr-serverless:GetDashboardForJobRun",
    "emr-serverless:AccessSystemProfileLogs",
    "glue:GetDatabases",
    "glue:SearchTables"
  ],
  "Resource": [
    "arn:aws:emr-serverless:region:account-id:/applications/applicationId/jobruns/jobid",
    "arn:aws:glue:region:account-id:catalog",
    "arn:aws:glue:region:account-id:database/*",
    "arn:aws:glue:region:account-id:table/*/*"
  ]
}
```

考量事項

對於在與任務相同的帳戶中存取 Lake Formation 中的資料庫或資料表的任務，可以看到用於偵錯的系統設定檔日誌。在下列案例中看不到它們：

- 如果使用 Lake Formation 許可管理的資料目錄具有跨帳戶資料庫和資料表
- 如果使用 Lake Formation 許可管理的資料目錄具有資源連結

使用 Glue Data Catalog 檢視

您可以在 Glue Data Catalog AWS 中建立和管理檢視，以搭配 EMR Serverless 使用。這些通常稱為 AWS Glue Data Catalog 檢視。這些檢視非常有用，因為它們支援多個 SQL 查詢引擎，因此您可以跨不同 AWS 服務存取相同的檢視，例如 EMR Serverless Amazon Athena 和 Amazon Redshift。

透過在 Data Catalog 中建立檢視，在中使用資源授予和標籤型存取控制 AWS Lake Formation 來授予其存取權。使用此存取控制方法，您不需要設定您在建立檢視時參考之資料表的額外存取權。這種授予許可的方法稱為定義者語意，這些檢視稱為定義者檢視。如需 Lake Formation 中存取控制的詳細資訊，請參閱 AWS Lake Formation 開發人員指南中的[授予和撤銷 Data Catalog 資源的許可](#)。

Data Catalog 檢視對於下列使用案例很有用：

- 精細的存取控制 – 您可以建立一個檢視，根據使用者需要的許可來限制資料存取。例如，您可使用 Data Catalog 中的視觀表阻止不在 HR 部門工作的員工查看個人身分識別資訊 (PII)。
- 完整檢視定義 – 透過在 Data Catalog 中的檢視上套用篩選條件，您可以確保 Data Catalog 中檢視中可用的資料記錄一律完整。
- 增強安全性 – 用於建立檢視的查詢定義必須已完成。此優點表示 Data Catalog 中的檢視較不容易受到惡意執行者的 SQL 命令影響。
- 簡單共用資料 – 與其他 AWS 帳戶共用資料而不移動資料。如需詳細資訊，請參閱[Lake Formation 中的跨帳戶資料共用](#)。

建立 Data Catalog 檢視

建立 Data Catalog 檢視的方法有多種。這包括使用 AWS CLI 或 Spark SQL。以下是幾個範例。

Using SQL

以下示範建立 Data Catalog 檢視的語法。請記下 MULTI DIALECT 檢視類型。這會將 Data Catalog 檢視與其他檢視區分開來。SECURITY 述詞指定為 DEFINER。這表示具有 DEFINER 語意的 Data Catalog 檢視。

```
CREATE [ OR REPLACE ] PROTECTED MULTI DIALECT VIEW [IF NOT EXISTS] view_name
[(column_name [COMMENT column_comment], ...) ]
[ COMMENT view_comment ]
[TBLPROPERTIES (property_name = property_value, ... )]
SECURITY DEFINER
AS query;
```

以下是遵循語法的範例CREATE陳述式：

```
CREATE PROTECTED MULTI DIALECT VIEW catalog_view
SECURITY DEFINER
AS
SELECT order_date, sum(totalprice) AS price
FROM source_table
GROUP BY order_date
```

您也可以使用 SQL 在試轉模式下建立檢視，以測試檢視建立，而無需實際建立資源。使用此選項會產生「試執行」來驗證輸入，如果驗證成功，則傳回將代表檢視之 Glue AWS 資料表物件的 JSON。在此情況下，不會建立實際檢視。

```
CREATE [ OR REPLACE ] PROTECTED MULTI DIALECT VIEW view_name
SECURITY DEFINER
[ SHOW VIEW JSON ]
AS view-sql
```

Using the AWS CLI

Note

當您使用 CLI 命令時，不會剖析用來建立檢視的 SQL。這可能會導致建立檢視的案例，但查詢不成功。在建立檢視之前，請務必測試您的 SQL 語法。

您可以使用下列 CLI 命令來建立檢視：

```
aws glue create-table --cli-input-json '{
  "DatabaseName": "database",
  "TableInput": {
    "Name": "view",
    "StorageDescriptor": {
```

```

    "Columns": [
      {
        "Name": "col1",
        "Type": "data-type"
      },
      ...
      {
        "Name": "col_n",
        "Type": "data-type"
      }
    ],
    "SerdeInfo": {}
  },
  "ViewDefinition": {
    "SubObjects": [
      "arn:aws:glue:aws-region:aws-account-id:table/database/referenced-table1",
      ...
      "arn:aws:glue:aws-region:aws-account-id:table/database/referenced-tableN",
    ],
    "IsProtected": true,
    "Representations": [
      {
        "Dialect": "SPARK",
        "DialectVersion": "1.0",
        "ViewOriginalText": "Spark-SQL",
        "ViewExpandedText": "Spark-SQL"
      }
    ]
  }
}
}'

```

受支援的檢視操作

下列命令片段為您顯示使用 Data Catalog 檢視的各種方式：

- 建立檢視

建立 data-catalog 檢視。以下是用於顯示從現有資料表建立檢視的範例：

```

CREATE PROTECTED MULTI DIALECT VIEW catalog_view
SECURITY DEFINER AS SELECT * FROM my_catalog.my_database.source_table

```

• 更改檢視

可用的語法：

- ALTER VIEW view_name [FORCE] ADD DIALECT AS query
- ALTER VIEW view_name [FORCE] UPDATE DIALECT AS query
- ALTER VIEW view_name DROP DIALECT

您可以使用 FORCE ADD DIALECT 選項，根據新的引擎方言強制更新結構描述和子物件。請注意，如果您不使用 FORCE 來更新其他引擎方言，則執行此操作可能會導致查詢錯誤。以下示範範例：

```
ALTER VIEW catalog_view FORCE ADD DIALECT
AS
SELECT order_date, sum(totalprice) AS price
FROM source_table
GROUP BY orderdate;
```

以下示範如何變更檢視以更新方言：

```
ALTER VIEW catalog_view UPDATE DIALECT AS
SELECT count(*) FROM my_catalog.my_database.source_table;
```

• 描述檢視

描述檢視的可用語法：

- SHOW COLUMNS {FROM|IN} view_name [{FROM|IN} database_name] – 如果使用者具有描述檢視所需的 AWS Glue 和 Lake Formation 許可，他們可以列出資料欄。以下示範顯示欄的幾個範例命令：

```
SHOW COLUMNS FROM my_database.source_table;
SHOW COLUMNS IN my_database.source_table;
```

- DESCRIBE view_name – 如果使用者具有描述檢視所需的 AWS Glue 和 Lake Formation 許可，他們可以列出檢視中的資料欄及其中繼資料。
- 捨棄檢視

可用的語法：

- DROP VIEW [IF EXISTS] view_name

下列範例顯示了 DROP 陳述式，在捨棄檢視之前測試檢視是否存在：

```
DROP VIEW IF EXISTS catalog_view;
```

- 顯示建立檢視

- SHOW CREATE VIEW view_name – 顯示用於建立指定檢視的 SQL 陳述式。以下是用於顯示建立資料目錄檢視的範例：

```
SHOW CREATE TABLE my_database.catalog_view;
CREATE PROTECTED MULTI DIALECT VIEW my_catalog.my_database.catalog_view (
  net_profit,
  customer_id,
  item_id,
  sold_date)
TBLPROPERTIES (
  'transient_lastDdlTime' = '1736267222')
SECURITY DEFINER AS SELECT * FROM
my_database.store_sales_partitioned_1f WHERE customer_id IN (SELECT customer_id
from source_table limit 10)
```

- 顯示檢視

列出目錄中的所有檢視，例如非正規檢視、多方位檢視 (MDV) 和不含 Spark 方言的 MDV。可用的語法如下：

- SHOW VIEWS [{ FROM | IN } database_name] [LIKE regex_pattern]:

以下示範顯示檢視的範例命令：

```
SHOW VIEWS IN marketing_analytics LIKE 'catalog_view*';
```

如需建立和設定資料型錄檢視的詳細資訊，請參閱《AWS Lake Formation 開發人員指南》中的[建置 AWS Glue 資料型錄檢視](#)。

查詢 Data Catalog 檢視

建立 Data Catalog 檢視後，您可以使用已啟用 AWS Lake Formation 精細存取控制的 Amazon EMR Serverless Spark 任務來查詢它。任務執行時間角色必須具有 Data Catalog 檢視上的 Lake Formation SELECT 許可。您不需要授予檢視中參考的基礎資料表存取權。

完成所有設定之後，您可以查詢檢視。例如，在 EMR Studio 中建立 EMR Serverless 應用程式後，請執行下列查詢來存取檢視。

```
SELECT * from my_database.catalog_view LIMIT 10;
```

實用的函數是 `invoker_principal`。它會傳回 EMRS 任務執行時間角色的唯一識別符。這可用於根據調用主體來控制檢視輸出。您可以使用它在檢視中新增條件，根據呼叫角色來精簡查詢結果。任務執行期角色必須具有 IAM `LakeFormation:GetDataLakePrincipal` 動作的許可，才能使用此函數。

```
select invoker_principal();
```

例如，您可以將此函數新增至 WHERE 子句，以精簡查詢結果。

考量和限制

當您建立 Data Catalog 檢視時，會套用下列條件：

- 您只能使用 Amazon EMR 7.6 及更高版本建立 Data Catalog 檢視。
- Data Catalog 檢視定義者必須具有對檢視存取的基礎基本資料表的 SELECT 存取權限。如果特定基本資料表對定義者角色施加了任何 Lake Formation 篩選條件，則建立 Data Catalog 檢視會失敗。
- 基礎資料表不得具有 Lake Formation 中的 `IAMAllowedPrincipals` 資料湖許可。如果存在，則錯誤多方言檢視只能參考沒有 `IAMAllowedPrincipals` 許可的資料表。
- 資料表的 Amazon S3 位置必須註冊為 Lake Formation 資料湖位置。如果未註冊資料表，則錯誤多方位檢視可能只會參考 Lake Formation 受管資料表。如需有關如何在 Lake Formation 中註冊 Amazon S3 位置的資訊，請參閱《AWS Lake Formation 開發人員指南》中的 [註冊 Amazon S3 位置](#)。
- 您只能建立 PROTECTED Data Catalog 檢視。不支援 UNPROTECTED 檢視。
- 您無法在 Data Catalog 檢視定義中參考另一個 AWS 帳戶中的資料表。您也無法在位於不同區域的相同帳戶中參考資料表。
- 若要跨帳戶或區域共用資料，整個檢視必須使用 Lake Formation 資源連結跨帳戶和跨區域共用。
- 不支援使用者定義的函數 (UDF 或 UDAF)。
- 您可以根據 Iceberg 資料表使用檢視。也支援開放資料表格式 Apache Hudi 和 Delta Lake。
- 您無法在 Data Catalog 檢視中參考其他檢視。
- AWS Glue Data Catalog 檢視結構描述一律使用小寫儲存。例如，如果您使用 DDL 陳述式來建立名為 `Castle` 的資料欄的 Glue Data Catalog 檢視 `Castle`，則在 Glue Data Catalog 中建立的資料欄將小寫為 `castle`。如果您接著將 DML 查詢中的資料欄名稱指定為 `Castle` 或 `CASTLE`，EMR Spark 會將名稱設為小寫，以便您執行查詢。但是，欄標題會使用您在查詢中指定的大小寫顯示。

如果您希望在 DML 查詢中指定的資料欄名稱與 Glue Data Catalog 中的資料欄名稱不相符的情況下，查詢失敗，請設定 `spark.sql.caseSensitive=true`。

支援開放式資料表格式

EMR Serverless 支援 Apache Hive、Apache Iceberg、Delta Lake (7.6.0+) 和 Apache Hudi (7.6.0+) 上的 SELECT 查詢。從 EMR 7.12 開始，Apache Hive、Apache Iceberg 和 Delta Lake 資料表支援修改資料表資料的 DML 和 DDL 操作，並使用 Lake Formation 提供的憑證。

考量和限制

一般

搭配 EMR Serverless 使用 Lake Formation 時，請檢閱下列限制。

Note

當您在 EMR Serverless 上為 Spark 任務啟用 Lake Formation 時，任務會啟動系統驅動程式和使用者驅動程式。如果您在啟動時指定了預先初始化的容量，驅動程式會從預先初始化的容量佈建，且系統驅動程式的數量等於您指定的使用者驅動程式數量。如果您選擇隨需容量，EMR Serverless 會啟動使用者驅動程式以外的系統驅動程式。若要估算與具有 Lake Formation 的 EMR Serverless 任務相關聯的成本，請使用 [AWS 定價計算工具](#)。

- Amazon EMR Serverless with Lake Formation 適用於所有支援的 [EMR Serverless 區域](#)。
- 啟用 Lake Formation 的應用程式不支援使用 [自訂 EMR Serverless 映像](#)。
- 您無法關閉 Lake Formation DynamicResourceAllocation 任務。
- 只能將 Lake Formation 與 Spark 任務搭配使用。
- 搭配 Lake Formation 的 EMR Serverless 僅支援整個任務的單一 Spark 工作階段。
- EMR Serverless with Lake Formation 僅支援透過資源連結共用的跨帳戶資料表查詢。
- 不支援下列內容：
 - 彈性分散式資料集 (RDD)
 - Spark 串流
 - 巢狀資料欄的存取控制

- EMR Serverless 會封鎖可能會破壞系統驅動程式完整隔離的功能，包括下列項目：
 - UDT、HiveUDF 以及任何涉及自訂類別的使用者定義的函數
 - 自訂資料來源
 - 為 Spark 延伸模組、連接器或中繼存放區提供額外的 jar
 - ANALYZE TABLE 命令
- 如果您的 EMR Serverless 應用程式位於具有 Amazon S3 VPC 端點的私有子網路中，且您連接端點政策來控制存取，則在任務將日誌資料傳送至 AWS Managed Amazon S3 之前，請將 VPC 政策中 [受管儲存](#) 中詳述的許可納入 S3 閘道端點。如需故障診斷請求，請聯絡 AWS 支援。
- 從 Amazon EMR 7.9.0 開始，Spark FGAC 在與 s3a:// 方案搭配使用時支援 S3AFileSystem。
- Amazon EMR 7.11 支援使用 CTAS 建立受管資料表。
- Amazon EMR 7.12 支援使用 CTAS 建立受管和外部資料表。

許可

- 為了強制執行存取控制，EXPLAIN PLAN 和 DDL 操作，例如 DESCRIBE TABLE 不會公開限制資訊。
- 當您向 Lake Formation 註冊資料表位置時，資料存取會使用 Lake Formation 儲存的登入資料，而不是 EMR Serverless 任務執行期角色的 IAM 許可。如果資料表位置的註冊角色設定錯誤，即使執行時間角色具有該位置的 S3 IAM 許可，任務也會失敗。
- 從 Amazon EMR 7.12 開始，您可以使用 DataFrameWriter (V2) 搭配附加模式中的 Lake Formation 憑證來寫入現有的 Hive 和 Iceberg 資料表。對於覆寫操作或建立新資料表時，EMR 會使用執行時間角色登入資料來修改資料表資料。
- 使用檢視或快取資料表做為來源資料時，適用下列限制（這些限制不適用於 AWS Glue Data Catalog 檢視）：
 - 對於 MERGE、DELETE 和 UPDATE 操作
 - 支援：使用檢視和快取資料表做為來源資料表。
 - 不支援：在指派和條件子句中使用檢視和快取資料表。
 - 對於建立或取代和取代資料表做為 SELECT 操作：
 - 不支援：使用檢視和快取資料表做為來源資料表。
- 只有在啟用刪除向量時，來源資料中具有 UDFs Delta Lake 資料表才支援 MERGE、DELETE 和 UPDATE 操作。

日誌和偵錯

- EMR Serverless 限制對已啟用 Lake Formation 之應用程式上的系統驅動程式 Spark 日誌的存取。由於系統驅動程式以更高的許可執行，因此系統驅動程式產生的事件和日誌可能包含敏感資訊。為了防止未經授權的使用者或程式碼存取此敏感資料，EMR Serverless 會停用對系統驅動程式日誌的存取。
- 系統設定檔日誌一律保留在受管儲存中 – 這是無法停用的強制性設定。這些日誌會使用客戶受管 KMS 金鑰或 AWS 受管 KMS 金鑰安全地儲存和加密。

Iceberg

使用 Apache Iceberg 時，請檢閱下列考量事項：

- 只能將 Apache Iceberg 與工作階段目錄搭配使用，不能與任意命名目錄搭配使用。
- 在 Lake Formation 中註冊的 Iceberg 資料表僅支援中繼資料資料表 history、metadata_log_entries、snapshots、files、manifests 和 refs。Amazon EMR 會隱藏可能具有敏感資料的資料欄，例如 partitions、path 和 summaries。此限制不適用於未在 Lake Formation 中註冊的 Iceberg 資料表。
- 未在 Lake Formation 中註冊的資料表支援所有 Iceberg 預存程序。所有資料表都不支援 register_table 和 migrate 程序。
- 我們建議您使用 Iceberg DataFrameWriterV2 而非 V1。

疑難排解

如需疑難排解方案，請參閱下列各節。

記錄

EMR Serverless 使用 Spark 資源描述檔來分割任務執行。EMR Serverless 使用使用者描述檔來執行您提供的程式碼，而系統描述檔會強制執行 Lake Formation 政策。您可以存取作為使用者設定檔執行之任務的日誌。

如需啟用偵錯 Lake Formation 任務的詳細資訊，請參閱[偵錯任務](#)。

Live UI 和 Spark History Server

Live UI 和 Spark History Server 具有從使用者設定檔產生的所有 Spark 事件，以及從系統驅動程式產生的修訂事件。

您可以在執行器索引標籤中查看使用者和系統驅動程式的所有任務。不過，日誌連結僅適用於使用者設定檔。此外，會從 Live UI 中修訂某些資訊，例如輸出記錄的數目。

任務失敗，Lake Formation 許可不足

請確保您的任務執行時期角色具有在您存取的資料表上執行 SELECT 和 DESCRIBE 的許可。

具有 RDD 執行的任務失敗

EMR Serverless 目前不支援已啟用 Lake Formation 任務的彈性分散式資料集 (RDD) 操作。

無法在 Amazon S3 中存取資料檔案

請確保已在 Lake Formation 中註冊資料湖的位置。

安全驗證例外狀況

EMR Serverless 偵測到安全驗證錯誤。如需協助，請聯絡 AWS 支援。

跨帳戶共用 AWS Glue Data Catalog 和資料表

可以跨帳戶共用資料庫和資料表，但仍使用 Lake Formation。如需詳細資訊，請參閱 [Lake Formation 中的跨帳戶資料共用](#) 和 [如何使用跨帳戶共用 AWS Glue Data Catalog 和資料表 AWS Lake Formation ?](#)。

Spark 原生精細存取控制允許列出的 PySpark API

為了維護安全性和資料存取控制，Spark 精細存取控制 (FGAC) 會限制某些 PySpark 函數。這些限制是透過下列方式強制執行：

- 阻止函數執行的明確封鎖
- 使函數無法運作的架構不相容
- 可能擲回錯誤、傳回存取遭拒訊息，或呼叫時不執行任何動作的函數

Spark FGAC 不支援下列 PySpark 功能：

- RDD 操作（使用 SparkRDDUnsupportedException 封鎖）
- Spark Connect（不支援）
- Spark 串流（不支援）

雖然我們已在原生 Spark FGAC 環境中測試列出的函數並確認它們如預期般運作，但我們的測試通常只涵蓋每個 API 的基本使用。具有多個輸入類型或複雜邏輯路徑的函數可能會有未經測試的案例。

對於此處未列出且未明確屬於上述不支援類別的任何函數，我們建議：

- 在 Gamma 環境或小型部署中先測試它們
- 在生產環境中使用它們之前驗證其行為

Note

如果您看到列出類別方法，但未列出其基本類別，則該方法仍應正常運作，這只是表示我們尚未明確驗證基本類別建構函數。

PySpark API 會組織成模組。下表詳細說明了每個模組內方法的一般支援。

模組名稱	狀態	備註
pyspark_core	支援	此模組包含主要 RDD 類別，大部分不支援這些函數。
pyspark_sql	支援	
pyspark_testing	支援	
pyspark_resource	支援	
pyspark_streaming	封鎖	Spark FGAC 中的串流用量遭到封鎖。
pyspark_mllib	實驗性	此模組包含以 RDD 為基礎的 ML 操作，而且這些函數大部分不受支援。此模組未經過徹底測試。
pyspark_ml	實驗性	此模組包含以 DataFrame 為基礎的 ML 操作，且大

模組名稱	狀態	備註
		部分支援這些函數。此模組未經過徹底測試。
pyspark_pandas	支援	
pyspark_pandas_slow	支援	
pyspark_connect	封鎖	Spark FGAC 中的 Spark Connect 用量遭到封鎖。
pyspark_pandas_connect	封鎖	Spark FGAC 中的 Spark Connect 用量遭到封鎖。
pyspark_pandas_slow_connect	封鎖	Spark FGAC 中的 Spark Connect 用量遭到封鎖。
pyspark_errors	實驗性	此模組未經過徹底測試。無法使用自訂錯誤類別。

API 允許清單

如需可下載且更容易搜尋的清單，可在[原生 FGAC 中允許的 Python 函數](#)中使用具有模組和類別的檔案。

工作者間加密

在 Amazon EMR 6.15.0 版及更高版本中，啟用 Spark 任務執行中工作者之間的交互 TLS 加密通訊。啟用時，EMR Serverless 會自動為任務執行下佈建的每個工作者產生並分配唯一的憑證。當這些工作者進行通訊以交換控制訊息或傳輸隨機播放資料時，他們會建立相互 TLS 連線，並使用設定的憑證來驗證彼此的身分。如果工作者無法驗證另一個憑證，TLS 交握會失敗，EMR Serverless 會中止它們之間的連線。

如果您使用 Lake Formation 搭配 EMR Serverless，則預設會啟用交互 TLS 加密。

在 EMR Serverless 上啟用交互 TLS 加密

若要在 Spark 應用程式上啟用交互 TLS 加密，請在[建立 EMR Serverless 應用程式](#)時將 `spark.ssl.internode.enabled` 設為 `true`。如果您使用 AWS 主控台建立

EMR Serverless 應用程式，請選擇使用自訂設定，然後展開應用程式組態，然後輸入您的 `runtimeConfiguration`。

```
aws emr-serverless create-application \  
--release-label emr-6.15.0 \  
--runtime-configuration '{  
  "classification": "spark-defaults",  
  "properties": {"spark.ssl.internode.enabled": "true"}  
}' \  
--type "SPARK"
```

如果您想要為個別 Spark 任務執行啟用交互 TLS 加密，請在使用時將 `spark.ssl.internode.enabled` 設定為 `truespark-submit`。

```
--conf spark.ssl.internode.enabled=true
```

使用 KMS CMK 進行磁碟加密

EMR Serverless 預設會使用服務擁有的加密金鑰來加密連接至工作者的所有磁碟。您可以選擇使用自己的 AWS KMS 客戶受管金鑰 (CMKs) 來加密這些磁碟。這可讓您進一步控制加密金鑰，包括建立和維護金鑰政策的能力，以及稽核金鑰用量。

您可以在建立應用程式或提交個別任務時設定磁碟加密。在應用程式層級啟用時，該應用程式上的所有任務都會繼承加密設定。您也可以提交任務時指定磁碟加密組態，覆寫應用程式的預設值。

Note

EMR Serverless 磁碟加密僅支援對稱 KMS 金鑰。不支援非對稱 KMS 金鑰。您必須使用在中建立的對稱加密 KMS 金鑰 AWS KMS。如需詳細資訊 AWS KMS，請參閱 [什麼是 AWS KMS ?](#)

使用加密內容

或者，EMR Serverless 使用加密內容，為加密操作提供額外的已驗證資料。加密內容是一組金鑰/值對，可包含非秘密的其他已驗證資料。加密內容以密碼編譯方式繫結至加密的資料，因此需要相同的加密內容才能解密資料。

在 EMR Serverless 中，您可以在設定磁碟加密時指定自訂加密內容。此加密內容包含在 AWS CloudTrail 日誌中，可協助您識別和了解 KMS 操作。

Note

請勿在加密內容中存放敏感資訊，因為它在 AWS CloudTrail 日誌中以純文字顯示。

使用客戶受管金鑰設定磁碟加密

CreateApplication

若要使用您自己的 KMS 金鑰加密磁碟，請在建立 EMR Serverless 應用程式時包含 `diskEncryptionConfiguration` 參數。

```
aws emr-serverless create-application \  
  --type TYPE \  
  --name APPLICATION_ID \  
  --release-label RELEASE_LABEL \  
  --region AWS_REGION \  
  --disk-encryption-configuration '{  
    "encryptionKeyArn": "key-arn",  
    "encryptionContext": {  
      "key": "value"  
    }  
  }'  
'
```

UpdateApplication

若要更新 KMS 金鑰 ARN 和/或加密內容，請在更新應用程式時以新值指定 `diskEncryptionConfiguration` 參數。

```
aws emr-serverless update-application \  
  --name APPLICATION_ID \  
  --region AWS_REGION \  
  --disk-encryption-configuration '{  
    "encryptionKeyArn": "key-arn",  
    "encryptionContext": {  
      "key": "value"  
    }  
  }'
```

```
}'
```

Note

若要在應用程式上取消設定設定的磁碟加密，請在更新應用程式 `diskEncryptionConfiguration` 期間傳遞空的。

StartJobRun

若要使用您自己的 KMS 金鑰加密磁碟，請在提交任務執行時使用 `diskEncryptionConfiguration` 組態。

```
--configuration-overrides '{
  "diskEncryptionConfiguration": {
    "encryptionKeyArn": "key-arn",
    "encryptionContext": {
      "key": "value"
    }
  }
}'
```

Public Livy 端點

若要在透過公有 Livy 端點建立 Spark 工作階段時，使用您自己的 KMS 金鑰加密磁碟，請在工作階段的 `conf` 物件中指定加密組態。

```
data = {
  "kind": "pyspark",
  "heartbeatTimeoutInSeconds": 60,
  "conf": {
    "emr-serverless.session.executionRoleArn": "role_arn",
    "spark.emr-serverless.disk.encryptionKeyArn": "key-arn",
    "spark.emr-serverless.disk.encryptionContext": "key1:value1,key2:value2" #
Optional
  }
}

# Send request to create a session with the Livy API endpoint
request = AWSRequest(method='POST', url=endpoint + "/sessions", data=json.dumps(data),
headers=headers)
```

磁碟加密所需的許可

EMR Serverless 的加密金鑰許可

當您使用自己的加密金鑰加密磁碟時，您必須為 `emr-serverless.amazonaws.com` 委託人設定下列 KMS 金鑰許可：

- `kms:GenerateDataKey`：產生用於加密磁碟區的資料金鑰
- `kms:Decrypt`：存取加密磁碟內容時解密資料金鑰

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "emr-serverless.amazonaws.com"
  },
  "Action": [
    "kms:Decrypt",
    "kms:GenerateDataKey"
  ],
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "aws:SourceArn": "arn:aws:emr-serverless:region:aws-account-id:/
applications/application-id"
    },
    "StringEquals": {
      "kms:EncryptionContext:applicationId": "application-id",
      "aws:SourceAccount": "aws-account-id"
    }
  }
}
```

作為安全最佳實務，建議您將 `aws:SourceArn` 條件金鑰新增至 KMS 金鑰政策。IAM 全域條件金鑰 `aws:SourceArn` 有助於確保 EMR Serverless 僅針對應用程式 ARN 使用 KMS 金鑰。此外，包括 `aws:SourceAccount` 條件金鑰，可將 KMS 金鑰的使用限制為來自條件中指定之 AWS 帳戶 ID 的請求，以提供另一層安全性。

任務執行期角色在其 IAM 政策中必須具有下列許可：

```
{
```

```

    "Sid": "Enable GDK and Decrypt",
    "Version": "2012-10-17",
    "Statement": {
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": "key-arn"
    }
  }
}

```

必要的使用者許可

提交任務的使用者必須具有使用金鑰的許可。您可以在 KMS 金鑰政策或使用者、群組或角色的 IAM 政策中指定許可。如果提交任務的使用者缺少 KMS 金鑰許可，EMR Serverless 會拒絕任務執行提交。

範例金鑰政策

下列金鑰政策提供 `kms:DescribeKey`、`kms:GenerateDataKey` 和 `kms:Decrypt` 的許可：

- `kms:DescribeKey`：在使用前，驗證客戶受管 KMS 金鑰已啟用和 SYMMETRIC。

```

{
  "Sid": "Enable DescribeKey",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/user-name"
  },
  "Action": [
    "kms:DescribeKey"
  ],
  "Resource": "*"
},
{
  "Sid": "Enable GDK and Decrypt",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/user-name"
  },
  "Action": [

```

```

    "kms:GenerateDataKey",
    "kms:Decrypt"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ViaService": "emr-serverless.region.amazonaws.com",
      "kms:EncryptionContext:key": "value"
    }
  }
}

```

作為安全最佳實務，建議您將 `kms:viaService` 條件金鑰新增至 KMS 金鑰政策。它將 KMS 金鑰的使用限制為僅來自 `emr-serverless` 的驗證請求。

IAM 政策範例

下列 IAM 政策提供許可給 `kms:DescribeKey`、`kms:GenerateDataKey` 和 `kms:Decrypt`。

```

{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:DescribeKey",
      "kms:GenerateDataKey",
      "kms:Decrypt"
    ],
    "Resource": "key-arn"
  }
}

```

監控金鑰用量

您可以監控 EMR Serverless through AWS CloudTrail. AWS CloudTrail captures 中客戶受管金鑰對的所有 API 呼叫 AWS KMS 做為事件的使用，包括來自 EMR Serverless 主控台、EMR Serverless API、AWS CLI 或 AWS SDK 的呼叫。

擷取的資訊包含您指定的加密內容，可協助您識別和稽核使用 KMS 金鑰的特定 EMR Serverless 資源。例如，您可能會在 中看到類似下列的事件 AWS CloudTrail。如需使用的詳細資訊 AWS CloudTrail，請參閱 [AWS CloudTrail 《使用者指南》](#)。

GenerateDataKey

EMR Serverless 正在建立加密磁碟區時 GenerateDataKey 操作的範例事件

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AWSService",
    "principalId": "user",
    "invokedBy": "AWS Internal"
  },
  "eventTime": "2025-07-28T21:43:51Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "GenerateDataKey",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "ipAddress",
  "userAgent": "userAgent",
  "requestParameters": {
    "encryptionContext": {
      "applicationId": "test"
    },
    "keyId": "arn:aws:kms:region:accountId:key/ffffffff-fffff-aaaaa-eeee-sample",
    "keySpec": "AES_256"
  },
  "responseElements": null,
  "additionalEventData": {
    "keyMaterialId":
"145c963debe558dfb01848d2a4539da940f3478852f86cfe2f52d5df796a5a02"
  },
  "requestID": "cc9d1c5e-97c4-4a4f-ae7a-e576sample",
  "eventID": "0b0fef09-f28d-4da8-a5a1-17b74sample",
  "readOnly": true,
  "resources": [
    {
      "accountId": "account",
      "type": "AWS::KMS::Key",
      "ARN": "arn:aws:kms:region:accountId:key/ffffffff-fffff-aaaaa-eeee-sample"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "accountId",
  "eventCategory": "Management"
}
```

```
}
```

解密

EMR Serverless 存取加密資料時解密操作的範例事件。

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AWSService",
    "principalId": "user",
    "invokedBy": "AWS Internal"
  },
  "eventTime": "2025-07-28T21:43:51Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "Decrypt",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "ipAddress",
  "userAgent": "userAgent",
  "requestParameters": {
    "encryptionContext": {
      "applicationId": "test"
    },
    "keyId": "arn:aws:kms:region:accountId:key/ffffffff-fffff-aaaaa-eeee-sample",
    "keySpec": "AES_256"
  },
  "responseElements": null,
  "additionalEventData": {
    "keyMaterialId":
    "145c963debe558dfb01848d2a4539da940f3478852f86cfe2f52d5df796a5a02"
  },
  "requestID": "cc9d1c5e-97c4-4a4f-ae7a-e576sample",
  "eventID": "0b0fef09-f28d-4da8-a5a1-17b74sample",
  "readOnly": true,
  "resources": [
    {
      "accountId": "account",
      "type": "AWS::KMS::Key",
      "ARN": "arn:aws:kms:region:accountId:key/ffffffff-fffff-aaaaa-eeee-sample"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
}
```

```
"recipientAccountId": "accountId",  
"eventCategory": "Management"  
}
```

進一步了解

下列資源會提供有關靜態資料加密的詳細資訊。

- 如需 AWS KMS 基本概念的詳細資訊，請參閱 [AWS KMS 開發人員指南](#)。
- 如需 安全最佳實務的詳細資訊 AWS KMS，請參閱 [AWS KMS 開發人員指南](#)。

使用 EMR Serverless 進行資料保護的 Secrets Manager

AWS Secrets Manager 是一種秘密儲存服務，用於保護資料庫登入資料、API 金鑰和其他秘密資訊。然後，在您的程式碼中，將硬式編碼憑證取代為對 Secrets Manager 的 API 呼叫。這有助於確保檢查程式碼的人員不會洩露秘密，因為秘密不存在。如需概觀，請參閱 [AWS Secrets Manager 使用者指南](#)。

Secrets Manager 會使用 AWS Key Management Service 金鑰加密秘密。如需詳細資訊，請參閱 AWS Secrets Manager 《使用者指南》中的 [秘密加密和解密](#)。

您可以設定 Secrets Manager 根據您指定的排程自動輪換秘密。這可讓您以短期秘密取代長期秘密，有助於大幅降低洩漏風險。如需詳細資訊，請參閱 AWS Secrets Manager 《使用者指南》中的 [輪換 AWS Secrets Manager 秘密](#)。

Amazon EMR Serverless 與 整合，AWS Secrets Manager 因此您可以將資料存放在 Secrets Manager 中，並在組態中使用秘密 ID。

EMR Serverless 如何使用秘密

當您將資料存放在 Secrets Manager 中並在 EMR Serverless 的組態中使用秘密 ID 時，不會以純文字將敏感組態資料傳遞給 EMR Serverless，並將其公開給外部 APIs。如果您指出金鑰值對包含存放在 Secrets Manager 中秘密的秘密 ID，EMR Serverless 會在將組態資料傳送給工作者以執行任務時擷取秘密。

若要指出組態的鍵值對包含存放在 Secrets Manager 中的秘密參考，請將 `EMR.secret@` 註釋新增至組態值。對於具有秘密 ID 註釋的任何組態屬性，EMR Serverless 會呼叫 Secrets Manager，並在任務執行時解析秘密。

如何建立秘密

若要建立秘密，請遵循AWS Secrets Manager 《使用者指南》中的[建立 AWS Secrets Manager 秘密](#)中的步驟。在步驟 3 中，選擇純文字欄位以輸入您的敏感值。

在組態分類中提供秘密

下列範例示範如何在的組態分類中提供秘密StartJobRun。如果您想要在應用程式層級設定 Secrets Manager 的分類，請參閱 [EMR Serverless 的預設應用程式組態](#)。

在範例中，將 *SecretName* 取代為要擷取的秘密名稱。如需詳細資訊，請參閱 [如何建立秘密](#)。

本節內容

- [指定秘密參考 - Spark](#)
- [指定秘密參考 - Hive](#)

指定秘密參考 - Spark

Example– 在 Spark 的外部 Hive 中繼存放區組態中指定秘密參考

```
aws emr-serverless start-job-run \  
  --application-id "application-id" \  
  --execution-role-arn "job-role-arn" \  
  --job-driver '{  
    "sparkSubmit": {  
      "entryPoint": "s3://amzn-s3-demo-bucket/scripts/spark-jdbc.py",  
      "sparkSubmitParameters": "--jars s3://amzn-s3-demo-bucket/mariadb-  
connector-java.jar  
      --conf  
      spark.hadoop.javax.jdo.option.ConnectionDriverName=org.mariadb.jdbc.Driver  
      --conf spark.hadoop.javax.jdo.option.ConnectionUserName=connection-user-  
name  
      --conf  
      spark.hadoop.javax.jdo.option.ConnectionPassword=EMR.secret@SecretName  
      --conf spark.hadoop.javax.jdo.option.ConnectionURL=jdbc:mysql://db-host:db-  
port/db-name  
      --conf spark.driver.cores=2  
      --conf spark.executor.memory=10G  
      --conf spark.driver.memory=6G  
      --conf spark.executor.cores=4"  
    }  
  }
```

```

}' \
--configuration-overrides '{
  "monitoringConfiguration": {
    "s3MonitoringConfiguration": {
      "logUri": "s3://amzn-s3-demo-bucket/spark/logs/"
    }
  }
}'

```

Example– 在 spark-defaults 分類中指定外部 Hive 中繼存放區組態的秘密參考

```

{
  "classification": "spark-defaults",
  "properties": {

    "spark.hadoop.javax.jdo.option.ConnectionDriverName": "org.mariadb.jdbc.Driver"
    "spark.hadoop.javax.jdo.option.ConnectionURL": "jdbc:mysql://db-host:db-port/db-name"
    "spark.hadoop.javax.jdo.option.ConnectionUserName": "connection-user-name"
    "spark.hadoop.javax.jdo.option.ConnectionPassword":
    "EMR.secret@SecretName",
  }
}

```

指定秘密參考 - Hive

Example– 在 Hive 的外部 Hive 中繼存放區組態中指定秘密參考

```

aws emr-serverless start-job-run \
--application-id "application-id" \
--execution-role-arn "job-role-arn" \
--job-driver '{
  "hive": {
    "query": "s3://amzn-s3-demo-bucket/emr-serverless-hive/query/hive-query.q1",
    "parameters": "--hiveconf hive.exec.scratchdir=s3://amzn-s3-demo-bucket/emr-
serverless-hive/hive/scratch
                  --hiveconf hive.metastore.warehouse.dir=s3://amzn-s3-demo-bucket/
emr-serverless-hive/hive/warehouse
                  --hiveconf javax.jdo.option.ConnectionUserName=username
                  --hiveconf
javax.jdo.option.ConnectionPassword=EMR.secret@SecretName
                  --hiveconf
hive.metastore.client.factory.class=org.apache.hadoop.hive.q1.metadata.SessionHiveMetaStoreCli

```

```

        --hiveconf
    javax.jdo.option.ConnectionDriverName=org.mariadb.jdbc.Driver
        --hiveconf javax.jdo.option.ConnectionURL=jdbc:mysql://db-host:db-
port/db-name"
    }
}' \
--configuration-overrides '{
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://amzn-s3-demo-bucket"
        }
    }
}'

```

Example– 在 hive-site 分類中指定外部 Hive 中繼存放區組態的秘密參考

```

{
    "classification": "hive-site",
    "properties": {
        "hive.metastore.client.factory.class":
"org.apache.hadoop.hive.q1.metastore.SessionHiveMetaStoreClientFactory",
        "javax.jdo.option.ConnectionDriverName": "org.mariadb.jdbc.Driver",
        "javax.jdo.option.ConnectionURL": "jdbc:mysql://db-host:db-port/db-name",
        "javax.jdo.option.ConnectionUserName": "username",
        "javax.jdo.option.ConnectionPassword": "EMR.secret@SecretName"
    }
}

```

授予 EMR Serverless 擷取秘密的存取權

若要允許 EMR Serverless 從 Secrets Manager 擷取秘密值，請在建立秘密時新增下列政策陳述式。您必須使用客戶管理的 KMS 金鑰建立秘密，EMR Serverless 才能讀取秘密值。如需詳細資訊，請參閱 AWS Secrets Manager 《使用者指南》中的 [KMS 金鑰的許可](#)。

在下列政策中，將 *applicationId* 取代為應用程式的 ID。

秘密的資源政策

您必須在中秘密的資源政策中包含下列許可 AWS Secrets Manager，以允許 EMR Serverless 擷取秘密值。為了確保只有特定應用程式可以擷取此秘密，您可以選擇在政策中指定 EMR Serverless 應用程式 ID 做為條件。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSECRETSMANAGERGetsecretvalue",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "emr-serverless.amazonaws.com"
        ]
      },
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceArn": "arn:aws:emr-serverless:*:123456789012:/applications/"
        }
      }
    }
  ]
}
```

使用下列 customer-managed AWS Key Management Service (AWS KMS) 金鑰政策建立您的秘密：
客戶受管 AWS KMS 金鑰的政策

```
{
  "Sid": "Allow EMR Serverless to use the key for decrypting secrets",
  "Effect": "Allow",
  "Principal": {
    "Service": [
      "emr-serverless.amazonaws.com"
    ]
  },
```

```
"Action": [
  "kms:Decrypt",
  "kms:DescribeKey"
],
"Resource": "*",
"Condition": {
  "StringEquals": {
    "kms:ViaService": "secretsmanager.AWS ##.amazonaws.com"
  }
}
}
```

輪換秘密

當您定期更新秘密時，輪換即為。您可以設定 AWS Secrets Manager，根據您指定的排程自動輪換秘密。如此一來，您可以將長期秘密取代為短期秘密。這有助於降低入侵的風險。當任務轉換為執行中狀態時，EMR Serverless 會從註釋組態擷取秘密值。如果您或程序更新 Secrets Manager 中的秘密值，則必須提交新任務，以便任務可以擷取更新的值。

Note

已處於執行中狀態的任務無法擷取更新的秘密值。這可能會導致任務失敗。

搭配 EMR Serverless 使用 Amazon S3 Access Grants

EMR Serverless 的 S3 Access Grants 概觀

透過 Amazon EMR 6.15.0 版及更高版本，Amazon S3 Access Grants 提供可擴展的存取控制解決方案，以增強從 EMR Serverless 存取 Amazon S3 資料的能力。如果您的 S3 資料具有複雜或大型的許可組態，請使用存取授權來擴展使用者、角色和應用程式的 S3 資料許可。

使用 S3 Access Grants 來增強對 Amazon S3 資料的存取，超出執行時間角色或連接到可存取 EMR Serverless 應用程式之身分的 IAM 角色授予的許可。

如需詳細資訊，請參閱 [《Amazon EMR 管理指南》](#) 中的 [使用 Amazon EMR 的 S3 存取授權管理存取](#)，以及 [《Amazon Simple Storage Service 使用者指南》](#) 中的 [使用 S3 存取授權管理存取](#)。

本節說明如何啟動使用 S3 Access Grants 提供 Amazon S3 中資料的存取權的 EMR Serverless 應用程式。如需了解透過其他 Amazon EMR 部署來使用 S3 Access Grants 的步驟，請參閱下列文件：

- [將 S3 Access Grants 與 Amazon EMR 搭配使用](#)
- [將 S3 Access Grants 與 Amazon EMR on EKS 搭配使用](#)

使用 S3 Access Grants 啟動 EMR Serverless 應用程式以進行資料管理

您可以在 EMR Serverless 上啟用 S3 存取授權，並啟動 Spark 應用程式。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 EMR Serverless 應用程式設定任務執行角色。包含您必須執行 Spark 任務並使用 S3 Access Grants `s3:GetDataAccess` 和 `s3:GetAccessGrantsInstanceForPrefix` 的必要 IAM 許可 `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

如果您為任務執行指定 IAM 角色，而該角色具有直接存取 S3 的額外許可，則使用者可以存取角色允許的資料，即使他們沒有 S3 存取授權的許可。

2. 啟動您的 EMR Serverless 應用程式，其 Amazon EMR 發行標籤為 6.15.0 或更高版本以及 `spark-defaults` 分類，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。

```
aws emr-serverless start-job-run \
  --application-id application-id \
  --execution-role-arn job-role-arn \
  --job-driver '{
    "sparkSubmit": {
      "entryPoint": "s3://us-east-1.elasticmapreduce/emr-containers/samples/wordcount/scripts/wordcount.py",
```

```
        "entryPointArguments": ["s3://amzn-s3-demo-destination-bucket1/wordcount_output"],
        "sparkSubmitParameters": "--conf spark.executor.cores=1 --conf spark.executor.memory=4g --conf spark.driver.cores=1 --conf spark.driver.memory=4g --conf spark.executor.instances=1"
    }
}' \
--configuration-overrides '{
  "applicationConfiguration": [{
    "classification": "spark-defaults",
    "properties": {
      "spark.hadoop.fs.s3.s3AccessGrants.enabled": "true",
      "spark.hadoop.fs.s3.s3AccessGrants.fallbackToIAM": "false"
    }
  ]
}]
}'
```

使用 EMR Serverless 的 S3 Access Grants 考量事項

如需將 Amazon S3 Access Grants 與 EMR Serverless 搭配使用時的重要支援、相容性和行為資訊，請參閱《Amazon [EMR 管理指南](#)》中的 [S3 Access Grants 與 Amazon EMR 的考量](#)。

使用 記錄 Amazon EMR Serverless API 呼叫 AWS CloudTrail

Amazon EMR Serverless 已與 整合 AWS CloudTrail，此服務可提供使用者、角色或 EMR Serverless 中 AWS 服務所採取之動作的記錄。CloudTrail 會將 EMR Serverless 的所有 API 呼叫擷取為事件。擷取的呼叫包括來自 EMR Serverless 主控台的呼叫，以及對 EMR Serverless API 操作的程式碼呼叫。如果您建立線索，請啟用 CloudTrail 事件持續交付至 Amazon S3 儲存貯體，包括 EMR Serverless 的事件。如果您未設定追蹤，您仍然可以在事件歷史記錄中存取 CloudTrail 主控台中的最新事件。您可以使用 CloudTrail 所收集的資訊，判斷向 EMR Serverless 提出的請求、提出請求的 IP 地址、提出請求的人員、提出請求的時間，以及其他詳細資訊。

若要進一步了解，請參閱[AWS CloudTrail 《使用者指南》](#)。

CloudTrail 中的 EMR Serverless 資訊

當您建立帳戶 AWS 帳戶 時，您的上會啟用 CloudTrail。當活動在 EMR Serverless 中發生時，該活動會與事件歷史記錄中的其他服務 AWS 事件一起記錄在 CloudTrail 事件中。您可以在 中存取、搜尋和下載最近的事件 AWS 帳戶。如需詳細資訊，請參閱[使用 CloudTrail 事件歷史記錄檢視事件](#)。

若要持續記錄中的事件 AWS 帳戶，包括 EMR Serverless 的事件，請建立追蹤。線索能讓 CloudTrail 將日誌檔案交付至 Amazon S3 儲存貯體。依預設，當您在主控台中建立追蹤時，該追蹤會套用至所有的 AWS 區域。線索會記錄 AWS 分割區中所有區域的事件，並將日誌檔案交付至您指定的 Amazon S3 儲存貯體。此外，設定其他 AWS 服務以進一步分析和處理 CloudTrail 日誌中收集的事件資料。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 EMR Serverless 動作，並記錄在 [EMR Serverless API 參考](#)中。例如，對 CreateApplication、StartJobRun 和 CancelJobRun 動作發出的呼叫會在 CloudTrail 記錄檔案中產生專案。

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

- 是否使用根或 AWS Identity and Access Management (IAM) 使用者登入資料提出請求。
- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 請求是否由其他 AWS 服務提出。

如需詳細資訊，請參閱 [CloudTrail userIdentity 元素](#)。

了解 EMR Serverless 日誌檔案項目

追蹤是一種組態，能讓事件以日誌檔案的形式交付到您指定的 Amazon S3 儲存貯體。CloudTrail 日誌檔案包含一或多個日誌專案。一個事件為任何來源提出的單一請求，並包含請求動作、請求的日期和時間、請求參數等資訊。CloudTrail 日誌檔並非依公有 API 呼叫的堆疊追蹤排序，因此不會以任何特定順序出現。

以下範例顯示的是展示 CreateApplication 動作的 CloudTrail 日誌項目。

```
{
  "eventVersion": "1.08",
  "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": {
        "creationDate": "2022-06-01T23:46:52Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2022-06-01T23:49:28Z",
  "eventSource": "emr-serverless.amazonaws.com",
  "eventName": "CreateApplication",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "PostmanRuntime/7.26.10",
  "requestParameters": {
    "name": "my-serverless-application",
    "releaseLabel": "emr-6.6",
    "type": "SPARK",
    "clientToken": "0a1b234c-de56-7890-1234-567890123456"
  },
  "responseElements": {
    "name": "my-serverless-application",
    "applicationId": "1234567890abcdef0",
    "arn": "arn:aws:emr-serverless:us-west-2:555555555555:/
applications/1234567890abcdef0"
  },
  "requestID": "890b8639-e51f-11e7-b038-EXAMPLE",
  "eventID": "874f89fa-70fc-4798-bc00-EXAMPLE",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "012345678910",
  "eventCategory": "Management"
}
```

Amazon EMR Serverless 的合規驗證

EMR Serverless 的安全性和合規性由第三方稽核人員評估為多個 AWS 合規計畫的一部分，包括下列項目：

- 系統和組織控制 (SOC)
- 支付卡產業資料安全標準 (PCI DSS)
- 聯邦風險與授權管理計畫 (FedRAMP) 中等
- 美國健康保險流通與責任法案 (HIPAA)

AWS 在合規計畫範圍內的 AWS 服務提供特定合規計畫範圍內的 [AWS 服務](#) 頻繁更新清單。

可使用 下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱 [以 AWS 成品下載報告](#)。

如需 AWS 合規計畫的詳細資訊，請參閱 [AWS 合規計畫](#)。

您使用 EMR Serverless 時的合規責任取決於資料的敏感度、組織的合規目標，以及適用的法律和法規。如果您使用 EMR Serverless 符合 HIPAA、PCI 或 FedRAMP Moderate 等標準，AWS 會提供資源來協助：

- [安全與合規快速入門指南](#)，討論部署以安全與合規為重心之基準環境的架構考量和步驟 AWS。
- [AWS 客戶合規指南](#) 可協助您透過合規的角度了解共同責任模型。這份指南橫跨多個架構 (包含國家標準技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準組織 (ISO))，總結保護 AWS 服務的最佳實務並將指引對應至安全控制。
- [AWS Config](#) 可用來評定資源組態與內部實務、業界準則和法規的合規狀態。
- [AWS 合規資源](#) 是工作手冊和指南的集合，可能適用於您的產業和位置。
- [AWS Security Hub](#) 可讓您全面檢視內的安全狀態，AWS 並協助您檢查是否符合安全產業標準和最佳實務。
- [AWS Audit Manager](#) – 這 AWS 服務 可協助您持續稽核 AWS 用量，以簡化您管理風險的方式，以及符合法規和業界標準的方式。

Amazon EMR Serverless 中的彈性

AWS 全球基礎設施是以 AWS 區域和可用區域為基礎建置。AWS 區域提供多個實體分隔和隔離的可用區域，這些可用區域以低延遲、高輸送量和高備援聯網連接。透過可用區域，設計和操作可在區域之

間自動容錯移轉的應用程式和資料庫，而不會中斷。可用區域的可用性、容錯能力和擴展能力，均較單一或多個資料中心的傳統基礎設施還高。

如需 AWS 區域和可用區域的詳細資訊，請參閱 [AWS 全球基礎設施](#)。

除了 AWS 全球基礎設施之外，Amazon EMR Serverless 透過 EMRFS 提供與 Amazon S3 的整合，以協助支援您的資料彈性和備份需求。

Amazon EMR Serverless 中的基礎設施安全性

Amazon EMR 是受管服務，受到 AWS 全球網路安全的保護。如需 AWS 安全服務以及如何 AWS 保護基礎設施的相關資訊，請參閱 [AWS 雲端安全](#)。若要使用基礎設施安全最佳實務來設計您的 AWS 環境，請參閱安全支柱 AWS Well-Architected Framework 中的 [基礎設施保護](#)。

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

- Transport Layer Security (TLS)。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 具備完美轉送私密(PFS)的密碼套件，例如 DHE (Ephemeral Diffie-Hellman)或 ECDHE (Elliptic Curve Ephemeral Diffie-Hellman)。現代系統(如 Java 7 和更新版本)大多會支援這些模式。

Amazon EMR Serverless 中的組態和漏洞分析

AWS 處理基本安全任務，例如訪客作業系統 (OS) 和資料庫修補、防火牆組態和災難復原。這些程序已由適當的第三方進行檢閱並認證。如需詳細資訊，請參閱下列資源：

- [Amazon EMR Serverless 的合規驗證](#)
- [共同的責任模型](#)
- [Amazon Web Services : 安全程序概觀](#)

的端點和配額 EMR Serverless

服務端點

若要以程式設計方式連線至 AWS 服務，您可以使用端點。端點是 AWS Web 服務的進入點 URL。除了標準 AWS 端點之外，有些 AWS 服務會在選取的區域中提供 FIPS 端點。下表列出 EMR Serverless 的服務端點。如需詳細資訊，請參閱 [AWS 服務端點](#)。

EMR Serverless 服務端點

區域名稱	區域	端點	通訊協定
美國東部 (俄亥俄)	us-east-2 (僅限於下列可用區域：use2-az1、use2-az2和 use2-az3)	emr-serverless.us-east-2.amazonaws.com	HTTPS
美國東部 (維吉尼亞北部)	us-east-1 (僅限於下列可用區域：use1-az1、use1-az2、use1-az5、use1-az4和 use1-az6)	emr-serverless.us-east-1.amazonaws.com emr-serverless-fips.us-east-1.amazonaws.com	HTTPS
美國西部 (加利佛尼亞北部)	us-west-1	emr-serverless.us-west-1.amazonaws.com	HTTPS
美國西部 (奧勒岡)	us-west-2	emr-serverless.us-west-2.amazonaws.com	HTTPS

區域名稱	區域	端點	通訊協定
		emr-serverless-fips.us-west-2.amazonaws.com	
非洲 (開普敦)	af-south-1	emr-serverless.af-south-1.amazonaws.com	HTTPS
亞太地區 (香港)	ap-east-1	emr-serverless.ap-east-1.amazonaws.com	HTTPS
亞太地區 (雅加達)	ap-southeast-3	emr-serverless.ap-southeast-3.amazonaws.com	HTTPS
亞太地區 (墨爾本)	ap-southeast-4	emr-serverless.ap-southeast-4.amazonaws.com	HTTPS
亞太地區 (馬來西亞)	ap-southeast-5	emr-serverless.ap-southeast-5.amazonaws.com	HTTPS

區域名稱	區域	端點	通訊協定
亞太地區 (孟買)	ap-south-1	emr-serverless.ap-south-1.amazonaws.com	HTTPS
亞太地區 (大阪)	ap-northeast-3	emr-serverless.ap-northeast-3.amazonaws.com	HTTPS
亞太地區 (首爾)	ap-northeast-2	emr-serverless.ap-northeast-2.amazonaws.com	HTTPS
亞太地區 (新加坡)	ap-southeast-1	emr-serverless.ap-southeast-1.amazonaws.com	HTTPS
亞太地區 (悉尼)	ap-southeast-2	emr-serverless.ap-southeast-2.amazonaws.com	HTTPS
亞太地區 (東京)	ap-northeast-1	emr-serverless.ap-northeast-1.amazonaws.com	HTTPS

區域名稱	區域	端點	通訊協定
加拿大 (中部)	ca-central-1 (僅限於下列可用區域：cac1-az1和cac1-az2)	emr-serverless.ca-central-1.amazonaws.com	HTTPS
加拿大西部 (卡加利)	ca-west-1	emr-serverless.ca-west-1.amazonaws.com	HTTPS
歐洲 (法蘭克福)	eu-central-1	emr-serverless.eu-central-1.amazonaws.com	HTTPS
歐洲 (蘇黎世)	eu-central-2	emr-serverless.eu-central-2.amazonaws.com	HTTPS
歐洲 (愛爾蘭)	eu-west-1	emr-serverless.eu-west-1.amazonaws.com	HTTPS
歐洲 (倫敦)	eu-west-2	emr-serverless.eu-west-2.amazonaws.com	HTTPS
歐洲 (米蘭)	eu-south-1	emr-serverless.eu-south-1.amazonaws.com	HTTPS

區域名稱	區域	端點	通訊協定
Europe (Paris)	eu-west-3	emr-serverless.eu-west-3.amazonaws.com	HTTPS
歐洲 (西班牙)	eu-south-2	emr-serverless.eu-south-2.amazonaws.com	HTTPS
歐洲 (斯德哥爾摩)	eu-north-1	emr-serverless.eu-north-1.amazonaws.com	HTTPS
以色列 (特拉維夫)	il-central-1	emr-serverless.il-central-1.amazonaws.com	HTTPS
Middle East (Bahrain)	me-south-1	emr-serverless.me-south-1.amazonaws.com	HTTPS
中東 (阿拉伯聯合大公國)	me-central-1	emr-serverless.me-central-1.amazonaws.com	HTTPS
南美洲 (聖保羅)	sa-east-1	emr-serverless.sa-east-1.amazonaws.com	HTTPS

區域名稱	區域	端點	通訊協定
中國 (北京)	cn-north-1 (僅限於下列可用區域 : cnn1-az1、cnn1-az2)	emr-serverless.cn-north-1.amazonaws.com.cn	HTTPS
AWS GovCloud (美國東部)	us-gov-east-1	emr-serverless.us-gov-east-1.amazonaws.com	HTTPS
AWS GovCloud (美國西部)	us-gov-west-1	emr-serverless.us-gov-west-1.amazonaws.com	HTTPS

Service Quotas

服務配額也稱為限制，是 AWS 帳戶 可以使用的服務資源或操作數量上限。EMR Serverless 每分鐘收集服務配額用量指標，並將其發佈在AWS/Usage命名空間中。

Note

新 AWS 帳戶的初始配額較低，可能會隨著時間增加。Amazon EMR Serverless 會監控每個內的帳戶用量 AWS 區域，然後根據您的用量自動增加配額。

下表列出 EMR Serverless 的服務配額。如需詳細資訊，請參閱[AWS 服務 配額](#)。

名稱	預設值限制	是否可調整？	Description
每個帳戶的並行 vCPUs 上限	16	是	目前 中帳戶可同時執行的 vCPUs 數目上限 AWS 區域。

API 限制

以下說明 每個區域的 API 限制 AWS 帳戶。

資源	預設配額
ListApplications	每秒 10 筆交易。每秒爆量 50 筆交易。
CreateApplication	每秒 1 筆交易。每秒暴增 25 次交易。
DeleteApplication	每秒 1 筆交易。每秒暴增 25 次交易。
GetApplication	每秒 10 筆交易。每秒爆量 50 筆交易。
UpdateApplication	每秒 1 筆交易。每秒暴增 25 次交易。
ListJobRuns	每秒 1 筆交易。每秒暴增 25 次交易。
StartJobRun	每秒 1 筆交易。每秒暴增 25 次交易。
GetDashboardForJobRun	每秒 1 筆交易。每秒暴增 2 次交易。
CancelJobRun	每秒 1 筆交易。每秒暴增 25 次交易。
GetJobRun	每秒 10 筆交易。每秒爆量 50 筆交易。
StartApplication	每秒 1 筆交易。每秒暴增 25 次交易。
StopApplication	每秒 1 筆交易。每秒暴增 25 次交易。

其他考量

下列清單包含 EMR Serverless 的其他考量事項。

• 下列提供 EMR Serverless AWS 區域：

- 美國東部 (俄亥俄)
- 美國東部 (維吉尼亞北部)
- 美國西部 (加利佛尼亞北部)
- 美國西部 (奧勒岡)
- 非洲 (開普敦)
- 亞太地區 (香港)
- 亞太區域 (台北)
- 亞太地區 (雅加達)
- 亞太地區 (孟買)
- 亞太地區 (海德拉巴)
- 亞太地區 (大阪)
- 亞太區域 (首爾)
- 亞太區域 (新加坡)
- 亞太地區 (雪梨)
- 亞太地區 (馬來西亞)
- 亞太區域 (紐西蘭)
- 亞太區域 (泰國)
- 亞太地區 (東京)
- 加拿大 (中部)
- 歐洲 (法蘭克福)
- 歐洲 (愛爾蘭)
- 歐洲 (倫敦)
- 歐洲 (米蘭)
- Europe (Paris)
- 歐洲 (西班牙)
- 歐洲 (斯德哥爾摩)

- Middle East (Bahrain)
- 中東 (阿拉伯聯合大公國)
- 墨西哥 (中部)
- 南美洲 (聖保羅)
- AWS GovCloud (美國東部)
- AWS GovCloud (美國西部)

如需與這些區域相關聯的端點清單，請參閱 [服務端點](#)。

- 任務執行的預設逾時為 12 小時。您可以使用 `startJobRun` API 或 AWS SDK 中的 `executionTimeoutMinutes` 屬性變更此設定。如果您希望任務執行永遠不會逾時，您可以將 `executionTimeoutMinutes` 設定為 0。例如，如果您有串流應用程式，請將 `executionTimeoutMinutes` 設定為 0，以允許串流任務持續執行。
- `getJobRun` API 中的 `billedResourceUtilization` 屬性會顯示 AWS 已針對任務執行計費的彙總 vCPU、記憶體和儲存體。計費資源包括工作者的 1 分鐘最低用量，以及每個工作者超過 20 GB 的額外儲存空間。這些資源不包含閒置預先初始化工作者的使用量。
- 如果沒有 VPC 連線，任務可以存取相同中的某些 AWS 服務端點 AWS 區域。這些服務包括 Amazon S3、AWS Glue、AWS Lake Formation、Amazon CloudWatch Logs、AWS KMS、AWS Security Token Service、Amazon DynamoDB 和 AWS Secrets Manager。您可以透過啟用 VPC 連線以存取其他 AWS 服務 [AWS PrivateLink](#)，但您不需要這麼做。若要存取外部服務，請使用 VPC 建立您的應用程式。
- EMR Serverless 不支援 HDFS。工作者上的本機磁碟是 EMR Serverless 在任務執行期間用來隨機播放和處理資料的暫時儲存體。

Amazon EMR Serverless 發行版本

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。每個版本都包含您選取的大數據應用程式、元件和功能，以便在執行任務時讓 Amazon EMR Serverless 部署和設定。

使用 Amazon EMR 6.6.0 及更高版本，部署 EMR Serverless。此部署選項不適用於舊版 Amazon EMR。當您提交任務時，請指定下列其中一個支援的版本。

主題

- [AWS runtime for Apache Spark \(emr-spark-8.0.0\)](#)
- [AWS runtime for Apache Spark \(emr-spark-8.0-預覽\)](#)
- [EMR Serverless 7.13.0](#)
- [EMR Serverless 7.12.0](#)
- [EMR Serverless 7.11.0](#)
- [EMR Serverless 7.10.0](#)
- [EMR Serverless 7.9.0](#)
- [EMR Serverless 7.8.0](#)
- [EMR Serverless 7.7.0](#)
- [EMR Serverless 7.6.0](#)
- [EMR Serverless 7.5.0](#)
- [EMR Serverless 7.4.0](#)
- [EMR Serverless 7.3.0](#)
- [EMR Serverless 7.2.0](#)
- [EMR Serverless 7.1.0](#)
- [EMR Serverless 7.0.0](#)
- [EMR Serverless 6.15.0](#)
- [EMR Serverless 6.14.0](#)
- [EMR Serverless 6.13.0](#)
- [EMR Serverless 6.12.0](#)
- [EMR Serverless 6.11.0](#)
- [EMR Serverless 6.10.0](#)

- [EMR Serverless 6.9.0](#)
- [EMR Serverless 6.8.0](#)
- [EMR Serverless 6.7.0](#)
- [EMR Serverless 6.6.0](#)

AWS runtime for Apache Spark (emr-spark-8.0.0)

下表列出 AWS runtime for Apache Spark(emr-spark-8.0.0) 可用的應用程式版本。

應用程式版本資訊

應用程式	版本
Spark	4.0.2-amzn-0
Iceberg	1.10.1-amzn-0
Delta	4.0.0-amzn-1-spark
Hudi	1.1.0-amzn-0

AWS runtime for Apache Spark (emr-spark-8.0.0) 版本備註

- GA 版本 – 這是AWS runtime for Apache Spark採用 Apache Spark 4.0.2 的一般發行版本。此版本適用於 EMR Serverless、EMR on EC2 和 EMR on EKS。
- 區域可用性 - 適用於提供 EMR Serverless 的所有 AWS 區域，中東（巴林）和中東（阿拉伯聯合大公國）區域除外。
- 已知限制 - 此版本不提供具有原生 FGAC 支援的 Spark Connect 安全端點。
- 其他文件 - 如需其他 Apache Spark 文件，請參閱 [Apache Spark 4.0.2 版本文件](#)。

開始使用

若要開始使用 Apache Spark 4.0.2，請使用 CLI AWS 建立 EMR Serverless 應用程式：

```
aws emr-serverless create-application --type SPARK \  
--release-label emr-spark-8.0.0 \  

```

```
--name spark4-serverless \
--region us-east-1
```

備註

- 此版本會取代預覽版本 (emr-spark-8.0-preview)。預覽僅限於 Spark 4.0.1，且缺少 FGAC、Hudi、資料連接器和持久性 Spark 歷史記錄伺服器。
- create-application 使用的 --type 參數 SPARK (大寫)。

AWS runtime for Apache Spark (emr-spark-8.0-預覽)

下表列出 AWS runtime for Apache Spark(emr-spark-8.0-preview) 可用的應用程式版本。

應用程式版本資訊

應用程式	版本
Spark	4.0.1-amzn-0

AWS runtime for Apache Spark (emr-spark-8.0-preview) 版本備註

- 預覽版本 – 這是AWS runtime for Apache Spark具有 Apache Spark 4.0.1 的預覽版本。此預覽僅適用於 EMR Serverless。
- 區域可用性 - 此預覽版本適用於所有可使用 EMR Serverless AWS 的區域，但中國和 AWS GovCloud (US) 區域除外。
- 應用程式版本資訊 - 此版本隨附下列應用程式版本：
 - AWS 適用於 Java 的 SDK 2.35.5, 1.12.792
 - Python 3.9、3.11, 3.12
 - Scala 2.13.16
 - AmazonCloudWatchAgent 1.300034.0-amzn-0
 - Delta 4.0.0-amzn-0-spark
 - Iceberg 1.10.0-amzn-spark-0
 - 對於支援 Corretto 17 (JDK 17) 的應用程式，此版本預設隨附 Amazon Corretto 17 (在 OpenJDK 上建置)。
- 預覽限制 - 此預覽版本不提供下列功能：

- 互動式和整合功能：不支援 SageMaker Unified Studio、EMR Studio 整合、Spark Connect、Livy 和 JupyterEnterpriseGateway。
- 資料表格式和存取控制：不支援具有資料列層級或資料欄層級篩選和 DDL/DML 運算子的 Hudi、Delta Universal Format 和精細存取控制 (FGAC)。
- 資料連接器：無法使用 spark-sql-kinesis、emr-dynamodb 和 spark-redshift 連接器。
- 歷史記錄伺服器：此預覽版本中無法使用持久性 Spark 歷史記錄伺服器。使用者仍然可以存取即時 Spark UI，以即時監控和偵錯作用中的無伺服器任務。
- 特殊功能：具體化視觀表不可用。
- 預覽功能 - 您可以在此預覽版本中測試下列功能。此預覽版本不建議用於生產工作負載：
 - SQL 功能：具有更嚴格類型處理的 ANSI SQL 模式、用於鏈結操作的 SQL PIPE 語法 (|>)、用於半結構化 JSON 資料的 VARIANT 資料類型、具有控制流程陳述式和工作階段變數的 SQL 指令碼，以及 SQL 使用者定義的函數。
 - 串流增強功能：具有 transformWithState 運算子的任意狀態處理 API v2、可查詢串流狀態的狀態資料來源讀取器（實驗性），以及具有改善 RocksDB 變更日誌檢查點的增強型狀態存放區。
 - 資料表格式支援：支援 VARIANT 資料類型的 Apache Iceberg v3、AWS S3 Tables 整合，以及適用於 Iceberg、Delta Lake 和 Hive 資料表 AWS Lake Formation 的 Full Table Access (FTA)。
- 其他文件 - 如需其他 Apache Spark 文件，請參閱 [Apache Spark 4.0.1 版本文件](#)。

開始使用

若要開始使用 Apache Spark 4.0.1 預覽版，請使用 CLI AWS 建立 EMR Serverless 應用程式：

```
aws emr-serverless create-application --type spark \
  --release-label emr-spark-8.0-preview \
  --region us-east-1 --name spark4-preview
```

EMR Serverless 7.13.0

下表列出 7.13.0 EMR Serverless 可用的應用程式版本。

應用程式	版本
Apache Spark	3.5.6
Apache Hive	3.1.3

應用程式	版本
Apache Tez	0.10.2

EMR Serverless 7.13.0 版本備註

- 新功能
 - 適用於互動式 PySpark 工作階段的 Spark Connect — EMR Serverless 現在支援透過 Apache Spark Connect 的互動式 PySpark 工作階段。透過 Amazon EMR 7.13.0 版和更新版本，您可以從本機 PySpark 用戶端連線至在 EMR Serverless 上執行的 Spark，包括 VS Code、PyCharm 和 Jupyter 筆記本等 IDEs。如需詳細資訊，請參閱[透過 Spark Connect 使用 Amazon EMR Serverless 執行互動式工作階段](#)。

EMR Serverless 7.12.0

下表列出 7.12.0 EMR Serverless 可用的應用程式版本。

應用程式	版本
Apache Spark	3.5.6
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.12.0 版本備註

- 新功能
 - 適用於 EMR Serverless 的無伺服器儲存 – Amazon EMR Serverless 推出無伺服器儲存，具有 EMR 7.12 版和更新版本，可消除 Apache Spark 工作負載的本機磁碟佈建。EMR Serverless 會自動處理中繼資料操作，例如不含儲存費用的隨機播放。無伺服器儲存會將儲存與運算分離，允許 Spark 在閒置時立即釋放工作者，而不是讓工作者保持作用中狀態以保留臨時資料。若要進一步了解，請參閱[無伺服器儲存](#)。
 - Iceberg 具體化視觀表 - 從 Amazon EMR 7.12.0 開始，Amazon EMR Spark 支援建立和管理 Iceberg 具體化視觀表 (MV)

- Hudi 完整資料表存取 - 從 Amazon EMR 7.12.0 開始，Amazon EMR 現在會根據 Lake Formation 中定義的政策，支援 Apache Spark 中 Apache Hudi 的完整資料表存取 (FTA) 控制。當任務角色具有完整資料表存取權時，此功能可在 Lake Formation 註冊的資料表上啟用來自 Amazon EMR Spark 任務的讀取和寫入操作。
- Iceberg 版本升級 - Amazon EMR 7.12.0 支援 Apache Iceberg 1.10 版

EMR Serverless 7.11.0

下表列出 7.11.0 EMR Serverless 可用的應用程式版本。

應用程式	版本
Apache Spark	3.5.6
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.11.0 版本備註

- 最大任務執行時間 – BATCH 任務 `executionTimeoutMinutesStartJobRun` 的作用中值從此版本開始為 7 天。 `executionTimeoutMinutes` 無法再設定為 0，即不會針對批次任務執行逾時。

EMR Serverless 7.10.0

下表列出 7.10.0 EMR Serverless 可用的應用程式版本。

應用程式	版本
Apache Spark	3.5.5
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.10.0 版本備註

- EMR Serverless 的指標 – 監控指標經過重組，以專注於 ApplicationName 和 JobName 維度。較舊的指標將不再更新，之後會繼續更新。如需詳細資訊，請參閱 [監控 EMR Serverless 應用程式和任務](#)。

EMR Serverless 7.9.0

下表列出 7.9.0 EMR Serverless 可用的應用程式版本。

應用程式	版本
Apache Spark	3.5.5
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.8.0

下表列出 7.8.0 EMR Serverless 可用的應用程式版本。

應用程式	版本
Apache Spark	3.5.4
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.7.0

下表列出 7.7.0 EMR Serverless 可用的應用程式版本。

應用程式	版本
Apache Spark	3.5.3
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.6.0

下表列出 7.6.0 EMR Serverless 可用的應用程式版本。

應用程式	版本
Apache Spark	3.5.3
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.5.0

下表列出 7.5.0 EMR Serverless 可用的應用程式版本。

應用程式	版本
Apache Spark	3.5.2
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.4.0

下表列出 7.4.0 EMR Serverless 可用的應用程式版本。

應用程式	版本
Apache Spark	3.5.2
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.3.0

下表列出 7.3.0 EMR Serverless 可用的應用程式版本。

應用程式	版本
Apache Spark	3.5.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.3.0 版本備註

- 使用 EMR Serverless 進行任務並行和佇列 – 當您在 Amazon EMR 7.3.0 版或更高版本上建立新的 EMR Serverless 應用程式時，預設會啟用任務並行和佇列。如需詳細資訊，請參閱 [the section called “任務並行和佇列”](#)，其中詳細說明了如何開始使用並行和佇列，也包含功能考量清單。

EMR Serverless 7.2.0

下表列出 7.2.0 EMR Serverless 可用的應用程式版本。

應用程式	版本
Apache Spark	3.5.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.2.0 版本備註

- Lake Formation 搭配 EMR Serverless – 您現在可以使用 AWS Lake Formation 在 S3 支援的 Data Catalog 資料表上套用精細存取控制。此功能可讓您設定 EMR Serverless Spark 任務中讀取查詢的資料表、資料列、資料欄和儲存格層級存取控制。如需詳細資訊，請參閱 [the section called “適用於 FGAC 的 Lake Formation”](#) 和 [the section called “考量和限制”](#)。

EMR Serverless 7.1.0

下表列出 7.1.0 EMR Serverless 可用的應用程式版本。

應用程式	版本
Apache Spark	3.5.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.0.0

下表列出 7.0.0 EMR Serverless 可用的應用程式版本。

應用程式	版本
Apache Spark	3.5.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.15.0

下表列出 6.15.0 EMR Serverless 可用的應用程式版本。

應用程式	版本
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.15.0 版本備註

- TLS 支援 – 使用 Amazon EMR Serverless 6.15.0 版及更高版本，在 Spark 任務執行中啟用工作者之間的交互 TLS 加密通訊。啟用時，EMR Serverless 會自動為其在任務執行下佈建的每個工作者產生唯一的憑證，供工作者在 TLS 交握期間用來驗證彼此身分，並建立加密通道以安全地處理資料。如需交互 TLS 加密的詳細資訊，請參閱[工作者間加密](#)。

EMR Serverless 6.14.0

下表列出 6.14.0 EMR Serverless 可用的應用程式版本。

應用程式	版本
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.13.0

下表列出 6.13.0 EMR Serverless 可用的應用程式版本。

應用程式	版本
Apache Spark	3.4.1
Apache Hive	3.1.3

應用程式	版本
Apache Tez	0.10.2

EMR Serverless 6.12.0

下表列出 6.12.0 EMR Serverless 可用的應用程式版本。

應用程式	版本
Apache Spark	3.4.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.11.0

下表列出 6.11.0 EMR Serverless 可用的應用程式版本。

應用程式	版本
Apache Spark	3.3.2
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.11.0 版本備註

- [存取其他帳戶中的 S3 資源](#) - 透過 6.11.0 版和更高版本，您可以設定多個 IAM 角色，以便在從 EMR Serverless 存取不同 AWS 帳戶中的 Amazon S3 儲存貯體時擔任。

EMR Serverless 6.10.0

下表列出 6.10.0 EMR Serverless 可用的應用程式版本。

應用程式	版本
Apache Spark	3.3.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.10.0 版本備註

- 對於 6.10.0 版或更新版本的 EMR Serverless 應用程式，`spark.dynamicAllocation.maxExecutors` 屬性的預設值為 `infinity`。舊版預設為 `100`。如需詳細資訊，請參閱 [Spark 任務屬性](#)。

EMR Serverless 6.9.0

下表列出 6.9.0 EMR Serverless 可用的應用程式版本。

應用程式	版本
Apache Spark	3.3.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.9.0 版本備註

- Apache Spark 的 Amazon Redshift 整合包含在 Amazon EMR 6.9.0 及更高版本中。以前是一個開放原始碼工具，本機整合是一個 Spark 連接器，可用於建置在 Amazon Redshift 和 Amazon Redshift Serverless 中讀取和寫入資料的 Apache Spark 應用程式。如需詳細資訊，請參閱 [在 Amazon EMR Serverless 上使用 Apache Spark 的 Amazon Redshift 整合](#)。
- EMR Serverless 6.9.0 版新增了 AWS Graviton2 (arm64) 架構的支援。您可以使用 `create-application` 和 `update-application` APIs 的 `architecture` 參數來選擇 arm64 架構。如需詳細資訊，請參閱 [Amazon EMR Serverless 架構選項](#)。

- 您現在可以直接從 EMR Serverless Spark 和 Hive 應用程式匯出、匯入、查詢和聯結 Amazon DynamoDB 資料表。如需詳細資訊，請參閱 [使用 Amazon EMR Serverless 連線至 DynamoDB](#)。

已知問題

- 如果您使用適用於 Apache Spark 的 Amazon Redshift 整合，並以 Parquet 格式具有微秒精確度的時間、時間戳記或時間戳記，連接器會將時間值四捨五入至最接近的毫秒值。請使用文字卸載格式 `unload_s3_format` 參數作為一種解決方法。

EMR Serverless 6.8.0

下表列出 6.8.0 EMR Serverless 可用的應用程式版本。

應用程式	版本
Apache Spark	3.3.0
Apache Hive	3.1.3
Apache Tez	0.9.2

EMR Serverless 6.7.0

下表列出 6.7.0 EMR Serverless 可用的應用程式版本。

應用程式	版本
Apache Spark	3.2.1
Apache Hive	3.1.3
Apache Tez	0.9.2

引擎特定的變更、增強功能和已解決的問題

下表列出新的引擎特定功能。

變更	說明
功能	Tez 排程器現在支援先佔 Tez 任務，而不是先佔容器

EMR Serverless 6.6.0

下表列出 6.6.0 EMR Serverless 可用的應用程式版本。

應用程式	版本
Apache Spark	3.2.0
Apache Hive	3.1.2
Apache Tez	0.9.2

EMR Serverless 初始版本備註

- EMR Serverless 支援 Spark 組態分類 spark-defaults。此分類會變更 Spark spark-defaults.conf XML 檔案中的值。組態分類可讓您自訂應用程式。如需詳細資訊，請參閱[設定應用程式](#)。
- EMR Serverless 支援 Hive 組態分類 hive-site、emrfs-site、tez-site 和 core-site。此分類可以分別變更 Hive hive-site.xml 檔案、Tez tez-site.xml 檔案、Amazon EMR 的 EMRFS 設定或 Hadoop core-site.xml 檔案中的值。組態分類可讓您自訂應用程式。如需詳細資訊，請參閱[設定應用程式](#)。

引擎特定的變更、增強功能和已解決的問題

- 下表列出 Hive 和 Tez 後端連接埠。

Hive 和 Tez 變更

變更	說明
向後移植	TEZ-4430 : 已修正 <code>tez.task.launch.cmd-opts</code> 屬性的問題
向後移植	HIVE-25971 : 已修正因開啟快取執行緒集區而導致的 Tez 任務關閉延遲

文件歷史記錄

下表說明自上次發行 EMR Serverless 以來文件的重要變更。如需有關此文件更新的詳細資訊，您可以訂閱 RSS 摘要。

變更	描述	日期
新的 GA 版本	AWS runtime for Apache Spark (emr-spark-8.0.0)	2026 年 5 月 21 日
新功能	使用 Spark Connect 執行互動式工作階段	2026 年 4 月 23 日
新的預覽版本	AWS runtime for Apache Spark (emr-spark-8.0-預覽)	2025 年 11 月 21 日
新版本	EMR Serverless 7.2.0	2024 年 7 月 25 日
新版本	EMR Serverless 7.1.0	2024 年 4 月 17 日
更新至現有政策。	將新的 SidCloudWatchPolicyStatement 和 EC2PolicyStatement 新增至 AmazonEMRServerlessServiceRolePolicy 政策。	2024 年 1 月 25 日
新版本	EMR Serverless 7.0.0	2023 年 12 月 29 日
新版本	EMR Serverless 6.15.0	2023 年 11 月 17 日
新功能	設定多個 IAM 角色，以便在從 EMR Serverless (6.11 及更高版本) 存取不同帳戶中的 Amazon S3 儲存貯體時擔任	2023 年 10 月 18 日
新版本	EMR Serverless 6.14.0	2023 年 10 月 17 日
新功能	EMR Serverless 的預設應用程式組態	2023 年 9 月 25 日

更新至預設 Hive 屬性	更新 <code>hive.driver.disk</code> 、 <code>hive.tez.disk.size</code> 、 <code>hive.tez.auto.reducer.parallelism</code> 和 <code>hive.tez.grouping.min-size</code> Hive 任務屬性的預設值 。	2023 年 9 月 12 日
新版本	EMR Serverless 6.13.0	2023 年 9 月 11 日
新版本	EMR Serverless 6.12.0	2023 年 7 月 21 日
新版本	EMR Serverless 6.11.0	2023 年 6 月 8 日
更新至服務連結角色政策	更新 AmazonEMRServerlessServiceRolePolicy SLR 角色以發佈 "AWS/Usage" 命名空間中的帳戶層級用量。	2023 年 4 月 20 日
EMR Serverless 一般可用性 (GA)	這是 EMR Serverless 的第一個公開版本。	2022 年 6 月 1 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。