Amazon EMR Serverless 使用者指南

Amazon EMR

Amazon EMR: Amazon EMR Serverless 使用者指南

Table of Contents

什麼是 Amazon EMR 無伺服器？

Amazon EMR 無伺服器是 Amazon EMR 提供無伺服器執行階段環境的部署選項。如此可簡化使用最新開放原始碼架構 (例如 Apache Spark 和 Apache Hive) 之分析應用程式的作業。有了EMR無伺服器，您不需要設定、最佳化、保護或操作叢集，就能使用這些架構執行應用程式。

EMR無伺服器可協助您避免資料處理工作的佈建過度或佈建不足的資源。EMR無伺服器會自動判斷應用程式所需的資源、取得這些資源來處理您的工作，並在工作完成時釋放資源。對於應用程式需要在幾秒鐘內回應的使用案例 (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。

有了EMR無伺服器，您將繼續獲得 Amazon 的優勢EMR，例如適用於熱門架構的開放原始碼相容性、並行性和最佳化執行時期效能。

EMR無伺服器適合想要使用開放原始碼架構輕鬆操作應用程式的客戶。它提供快速的作業啟動、自動容量管理，以及直接的成本控制。

概念

在本節中，我們涵蓋EMR無伺服器使用者指南中出現的EMR無伺服器術語和概念。

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，這些應用程式和功能可供EMR無伺服器部署和設定，以便它們能夠執行您的應用程式。建立應用程式時，您必須指定其發行版本。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。若要深入瞭解發行前版本，請參閱Amazon EMR 無伺服器發行版本。

應用程式

使用EMR無伺服器，您可以建立一或多個使用開放原始碼分析架構的EMR無伺服器應用程式。若要建立應用程式，您必須指定下列屬性：

- 您要使用的開EMR放原始碼架構版本的 Amazon 發行版本。若要判斷您的發行版本，請參閱Amazon EMR 無伺服器發行版本。
- 您希望應用程序使用的特定運行時，例如 Apache 星火或 Apache 蜂房。

建立應用程式之後，您可以將資料處理工作或互動式要求提交至應用程式。

每個EMR無伺服器應用程式都在安全的 Amazon Virtual Private Cloud (VPC) 上執行，與其他應用程式完全不同。此外，您可以使用 AWS Identity and Access Management (IAM) 定義哪些使用者和角色可以存取應用程式的原則。您也可以指定限制，以控制和追蹤應用程式產生的使用成本。

當您需要執行下列動作時，請考慮建立多個應用程式：

- 使用不同的開源框架
- 針對不同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
- 為測試和生產方案維護單獨的邏輯環境
- 為不同團隊提供獨立的邏輯環境，提供獨立的成本控制和使用情況
- 分隔不同的 line-of-business 應用

EMR無伺服器是一種區域性服務，可簡化工作負載在一個區域中跨多個可用區域執行的方式。若要進一步瞭解如何搭配EMR無伺服器使用應用程式，請參閱與應用程式互動。

作業執行

工作執行是提交至EMR無伺服器應用程式的要求，應用程式會以不同步方式執行並追蹤完成。工作範例包括您提交至 Apache Hive 應用程式的 HiveQL 查詢，或是您送出至 Apache Spark 應用程式的 PySpark 資料處理指令碼。當您提交工作時，您必須指定作業用來存取的執行時期角色 (編寫於IAM) AWS 資源，例如 Amazon S3 對象。您可以將多個工作執行請求提交至應用程式，而且每個工作執行都可以使用不同的執行時期角色來存取 AWS 的費用。EMR無伺服器應用程式在收到工作後立即開始執行工作，並同時執行多個工作要求。若要深入瞭解EMR無伺服器如何執行工作，請參閱執行任務。

工作程序

EMR無伺服器應用程式內部使用 Worker 來執行您的工作負載。這些 Worker 的預設大小取決於您的應用程式類型和 Amazon EMR 發行版本。當您排定工作執行時，您可以覆寫這些大小。

當您提交工作時，EMR無伺服器會計算應用程式對工作所需的資源，並排程工作者。EMR無伺服器可將您的工作負載分解為任務、下載影像、佈建和設定工作者，並在工作完成時將其解除委任。EMR無伺服器會根據工作每個階段所需的工作負載和平行處理原則，自動擴展或縮減工作者。這種自動擴展功能讓您無需預估應用程式執行工作負載所需的 Worker 數量。

預先初始化容量

EMR無伺服器提供預先初始化的容量功能，可讓 Worker 保持初始化，並在幾秒鐘內做好回應。這種能力有效地為應用程序創建了一個溫暖的工作人員池。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，工作可以立即啟動，以便您可以實作反覆應用程式和時間敏感的工作。若要進一步瞭解預先初始化的 Worker，請參閱設定應用程式。

EMR工作室

EMRStudio 是可用來管理EMR無伺服器應用程式的使用者主控台。當您建立第一個EMR無伺服器應用程式時，如果您的帳戶中沒有 EMR Studio，我們會自動為您建立一個工作室。您可以從 Amazon EMR 主控台存取 EMR Studio，也可以透過身分識別供應商 (IdP) IAM 或IAM身分中心開啟聯合存取。執行此操作時，使用者無需直接存取 Amazon EMR 主控台即可存取 Studio 和管理EMR無伺服器應用程式。若要深入了解EMR無伺服器應用程式如何與 EMR Studio 搭配運作，請參閱從 EMR Studio 主控台與您的應用程式互動和從 EMR Studio 主控台執行工作。

開始使用EMR無伺服器的先決條件

主題

註冊一個 AWS 帳戶

如果您沒有 AWS 帳戶，請完成下列步驟來建立一個步驟。

若要註冊成為 AWS 帳戶

1. 打開https://portal.aws.amazon.com/billing/註冊。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   當您註冊一個 AWS 帳戶，一個 AWS 帳戶根使用者已建立。根使用者可以存取所有 AWS 服務 和帳戶中的資源。作為安全最佳實務，請將管理存取權指派給使用者，並且僅使用根使用者來執行需要根使用者存取權的任務。

AWS 註冊過程完成後，會向您發送確認電子郵件。您可以隨時前往 https://aws.amazon.com/並選擇「我的帳戶」，檢視目前的帳戶活動並管理您的帳戶。

建立具有管理存取權的使用者

在您註冊一個 AWS 帳戶，保護您的 AWS 帳戶根使用者，啟用 AWS IAM Identity Center，並建立系統管理使用者，這樣您就不會將 root 使用者用於日常工作。

保護您的 AWS 帳戶根使用者

1. 登入 AWS Management Console通過選擇 Root 用戶並輸入您的帳戶所有者 AWS 帳戶 電子郵件地址。在下一頁中，輸入您的密碼。

如需使用 root 使用者登入的說明，請參閱以 root 使用者身分登入 AWS 登入 使用者指南。

2. 為您的 root 使用者開啟多因素驗證 (MFA)。

   如需指示，請參閱為您的MFA裝置啟用虛擬裝置 AWS 帳戶 使用者指南中的 root IAM 使用者 (主控台)。

建立具有管理存取權的使用者

1. 啟用IAM身分識別中心。

   如需指示，請參閱啟用 AWS IAM Identity Center 中的 AWS IAM Identity Center 使用者指南。

2. 在IAM身分識別中心中，將管理存取權授與使用者。

   若要取得有關使用 IAM Identity Center 目錄 做為您的身分識別來源，請參閱以預設值設定使用者存取 IAM Identity Center 目錄 中的 AWS IAM Identity Center 使用者指南。

以具有管理存取權的使用者身分登入

- 若要使用您的 IAM Identity Center 使用者登入URL，請使用建立IAM身分識別中心使用者時傳送至您電子郵件地址的登入資訊。

  如需使用IAM身分識別中心使用者登入的說明，請參閱登入 AWS 存取入口網站 AWS 登入 使用者指南。

指派存取權給其他使用者

1. 在 IAM Identity Center 中，建立遵循套用最低權限權限的最佳作法的權限集。

   如需指示，請參閱 AWS IAM Identity Center 使用者指南。

2. 將使用者指派至群組，然後對該群組指派單一登入存取權。

   如需指示，請參閱新增群組 AWS IAM Identity Center 使用者指南。

授予許可

在生產環境中，建議您使用更精細的原則。如需此類政策的範例，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，請參閱的存取管理 AWS 《IAM使用者指南》 中的資源。

對於需要在沙箱環境中開始使用EMR無伺服器的使用者，請使用類似下列內容的原則：

```
{
    "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。遵循中建立權限集中的指示 AWS IAM Identity Center 使用者指南。

- IAM透過身分識別提供者管理的使用者：

  建立聯合身分的角色。請遵循《使用指南》中的〈為第三方身分識別提供IAM者 (同盟) 建立角色〉中的指示進行。

- IAM使用者：
  - 建立您的使用者可擔任的角色。請按照《用戶指南》中的「為IAM用戶創建角色」中的IAM說明進行操作。
  - (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《使用指南》中的「向使用者 (主控台) 新增權限」IAM 中的指示進行。

授與程式設計存取權

如果用戶想要與之互動，則需要以程式設計方式存取 AWS 的之外 AWS Management Console。 授與程式設計存取權的方式取決於存取的使用者類型 AWS.

若要授與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	By
人力身分 (在IAM身分識別中心管理的使用者)	使用臨時登入資料來簽署程式設計要求 AWS CLI, AWS SDKs，或 AWS APIs.	請依照您要使用的介面所提供的指示操作。 • 對於 AWS CLI，請參閱配置 AWS CLI 若要使用 AWS IAM Identity Center 中的 AWS Command Line Interface 使用者指南。

哪個使用者需要程式設計存取權？	到	By
		• 用於 AWS SDKs、工具和 AWS APIs，請參閱IAM身分識別中心驗證 AWS SDKs和工具參考指南。
IAM	使用臨時登入資料來簽署程式設計要求 AWS CLI, AWS SDKs，或 AWS APIs.	遵循使用臨時認證中的說明進行操作 AWS《IAM使用者指南》中的資源。
IAM	(不建議使用) 使用長期認證來簽署程式設計要求 AWS CLI, AWS SDKs，或 AWS APIs.	請依照您要使用的介面所提供的指示操作。 • 對於 AWS CLI，請參閱「使用使用IAM者認證進行驗證」AWS Command Line Interface 使用者指南。 • 用於 AWS SDKs和工具，請參閱使用長期認證進行身份驗證 AWS SDKs和工具參考指南。 • 用於 AWS APIs，請參閱《使用指南》中的〈管理使IAM用IAM者的存取金鑰〉。

安裝和配置 AWS CLI

如果您想要使用EMR無伺服器APIs，您必須安裝最新版本的 AWS Command Line Interface (AWS CLI). 你不需要 AWS CLI 從 EMR Studio 主控台使用EMR無伺服器，您可以按照中的CLI步驟在從主控台開始使用 EMR Serverless不使用.

若要設定 AWS CLI

1. 若要安裝最新版本的 AWS CLI 如需 macOS、Linux 或視窗，請參閱安裝或更新最新版本的 AWS CLI.

2. 若要設定 AWS CLI 並安全設置您的訪問 AWS 服務，包括EMR無伺服器，請參閱使aws configure用快速設定。
3. 若要驗證設定，請在 DataBrew 命令提示字元中輸入下列命令。

```
aws emr-serverless help
```

AWS CLI 指令使用預設 AWS 區域 從您的配置中，除非您使用參數或配置文件進行設置。若要設定 AWS 區域 使用參數，您可以將--region參數添加到每個命令中。

若要設定 AWS 區域 使用配置文件，首先在~/.aws/config文件或文件中添加一個命名的配置%UserProfile%/.aws/config文件（適用於 Microsoft Windows）。依照 [已命名] 設定檔中的步驟執行 AWS CLI。 接下來，設置 AWS 區域 和其他設定的指令類似於下列範例中的指令。

```
[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

本節中大多數以主控台為導向的主題都是從 Amazon 主控台開始。EMR如果您尚未登入 AWS 帳戶，登錄，然後打開 Amazon EMR 控制台並繼續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多數部分顯示預設選項。

啟動 EMR Serverless 應用程式之前，請完成下列任務。

主題

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 Serverless EMR 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。

為無EMR伺服器準備儲存體

在本教學課程中，您將使用 S3 儲存貯體來儲存使用 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。使用新建立的儲存貯體*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EMR過 Studio 以EMR無伺服器執行互動式工作負載。

建立任務執行期角色

EMR Serverless 中執行的任務會使用執行期角色，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 儲存貯體託管資料和指令碼。儲存貯*amzn-s3-demo-bucket*體存放輸出。

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。接下來，將所需的 S3 存取政策連接至該角色。下列步驟會引導您完成整個程序。

Console

1. 在 導覽至IAM主控台https://console.aws.amazon.com/iam/。
2. 在左側導覽窗格中，選擇 Roles (角色)。
3. 選擇建立角色。
4. 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 。

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

5. 選擇下一步導覽至新增許可頁面，然後選擇建立政策 。
6. 建立新索引標籤上會開啟建立政策頁面。貼上JSON以下政策。

> **Important**
>
> 將下列政策*amzn-s3-demo-bucket*中的 取代為在 中建立的實際儲存貯體名稱為無EMR伺服器準備儲存體。這是 S3 存取的基本政策。如需更多任務執行期角色範例，請參閱 Amazon EMR Serverless 的任務執行期角色。

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

7. 在檢閱政策頁面上，輸入政策的名稱，例如 EMRServerlessS3AndGlueAccessPolicy。
8. 重新整理連接許可政策頁面，然後選擇 EMRServerlessS3AndGlueAccessPolicy。
9. 在名稱中，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

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

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

2. 建立名為 IAM的角色EMRServerlessS3RuntimeRole。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
```

請記下輸出中的 ARN。您可以在任務提交期間使用新角色ARN的 ，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IAM的政策。這提供對存放在公有 S3 儲存貯體中的指令碼和資料的讀取存取權，以及對 的讀取寫入存取權*amzn-s3-demo-bucket*。

> **Important**
>
> 將下列政策*amzn-s3-demo-bucket*中的 取代為在 中建立的實際儲存貯體名稱為無EMR伺服器準備儲存體。這是 AWS Glue 和 S3 存取的基本政策。如需更多任務執行期角色範例，請參閱 Amazon EMR Serverless 的任務執行期角色。

```
{
    "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",Understanding default application behavior,
 including auto-start and auto-stop, as well as maximum capacity and worker
 configurations for configuring an application with &EMRServerless;.
                "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IAM的政策檔案建立名為EMRServerlessS3AndGlueAccessPolicy的政策。 請注意輸出ARN中的 ，因為您將在下一個步驟中使用新政策ARN的 。

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

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

5. 將IAM政策連接至EMRServerlessS3AndGlueAccessPolicy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

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：檢視應用程式使用者介面和日誌
- 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 Management Console 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無伺服器以導覽至無EMR伺服器登陸頁面。
3. 若要建立或管理無EMR伺服器應用程式，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，以便您可以建立和管理應用程式。
4. 在在新索引標籤中開啟的建立 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，在啟用互動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，請參閱建立工作室。
5. 選取建立應用程式以建立第一個應用程式。

繼續前往下一節步驟 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，我們會使用 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。

執行 Spark 任務

1. 使用下列命令將範例指令碼上傳至`wordcount.py`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. 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。在這裡，選擇提交任務選項。
3. 在提交任務頁面上，完成下列各項。
   - 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`作為 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

在教學課程的此部分中，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，請先建立包含所有 Hive 查詢的檔案，以作為單一任務的一部分執行，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 的檔案`hive-query.ql`，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.ql`上傳至 S3 儲存貯體。

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

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。在這裡，選擇提交任務選項。
4. 在提交任務頁面上，完成下列各項。
   - 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.ql`作為 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

使用 Amazon EMR 6.14.0 及更高版本，您可以使用 EMR Studio 中託管的筆記本，為 Spark in EMR Serverless 執行互動式工作負載。如需包含許可和先決條件的詳細資訊，請參閱 透EMR過 Studio 以EMR無伺服器執行互動式工作負載。

建立應用程式並設定必要的許可後，請使用下列步驟使用 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工作室按鈕。
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 作為運算類型 。
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，輸入無EMR伺服器應用程式可以為任務執行擔任IAM的角色名稱。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。您也可以在 Serverless EMR 上瀏覽範例筆記本，並將其複製到工作區。若要存取筆記本範例，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 是提供指標以監控任務的即時介面。如需詳細資訊，請參閱 Amazon EMR Serverless 使用者指南 中的監控無伺服器應用程式和任務。 EMR

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

步驟 3：檢視應用程式使用者介面和日誌

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，但仍建議您發行您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 Spark 任務和 Hive 任務。

從 開始 AWS CLI

步驟 1：建立 EMR Serverless 應用程式

使用 `emr-serverless create-application`命令來建立第一個 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。應用程式的名稱為選用。

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 啟動應用程式，並在任務提交期間，在此之後稱為 *application-id*。

在您繼續前往 之前步驟 2：將任務執行提交至您的 EMR Serverless 應用程式，請確定您的應用程式已使用 get-application 達到 CREATED 狀態API。

```
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，以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，請參閱設定應用程式。

步驟 2：將任務執行提交至您的 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 執行期傳送至您建立的 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我們將執行到新儲存貯體的範例指令碼。

   ```
   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* 以ARN您在 中建立的執行期角色取代 建立任務執行期角色。替代 *job-run-name* 使用您要呼叫任務執行的名稱。使用您建立的 Amazon S3 儲存貯體取代所有*amzn-s3-demo-bucket*字串，然後/output新增至路徑。這會在儲存貯體中建立新的資料夾，而 EMR Serverless 可以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"
        }
    }'
```

3. 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，請先建立包含所有 Hive 查詢的檔案，以作為單一任務的一部分執行，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 的檔案`hive-query.ql`，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.ql`上傳至 S3 儲存貯體。

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

3. 在下列命令中，*application-id*使用您自己的應用程式 ID 取代 。*job-role-arn* 以ARN您在 中建立的執行期角色取代 建立任務執行期角色。使用您建立的 Amazon S3 儲存貯體取代所有*amzn-s3-demo-bucket*字串，並將 `/output`和 新增至`/logs`路徑。這會在儲存貯體中建立新的資料夾，無EMR伺服器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.ql",
          "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 時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，EMRServerless 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由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 時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，EMRServerless 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

完成本教學課程的使用後，請考慮刪除您建立的資源。建議您發行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* 以 中建立的 S3 儲存貯體實際名稱取代 為無EMR伺服器準備儲存體。

```
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，請將政策與角色分離。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 Spark 任務和 Hive 任務。

與應用程式互動

本節說明如何與 Amazon EMR Serverless 應用程式與 互動， AWS CLI 以及 Spark 和 Hive 引擎的預設值。

主題

應用程式狀態

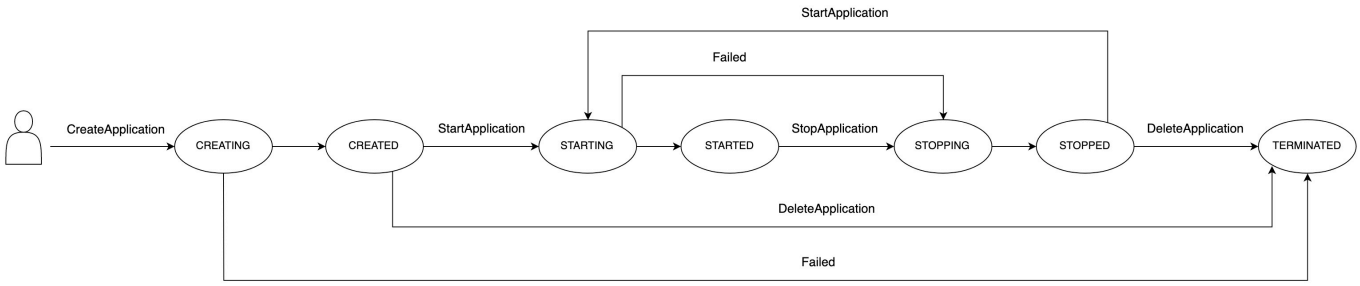
當您使用 EMR Serverless 建立應用程式時，應用程式執行會進入 `CREATING` 狀態。工作將經過以下狀態，直到其失敗 (以 `0` 代碼結束) 或失敗 (以與非零代碼結束) 為止。

應用程式可以具有下列狀態：

州	描述
正在建立	應用程式正在準備中，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。
已啟動	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。

州	描述
已停止	應用程式已停止，而且應用程式上沒有正在執行的資源。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，且不會顯示在您的應用程式清單中。

下圖顯示 EMR Serverless 應用程式狀態的軌跡。



從 EMR Studio 主控台與您的應用程式互動

從 EMR Studio 主控台，您可以建立、檢視和管理無EMR伺服器應用程式。若要導覽至 EMR Studio 主控台，請遵循從主控台 開始使用中的指示。

建立應用程式

使用建立應用程式頁面，您可以依照下列步驟建立無EMR伺服器應用程式。

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. 在版本欄位中，選擇EMR版本號碼。
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。

   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 Spark 驅動程式和一個執行器，以及一個 Hive 驅動程式和一個 Tez 任務。預設設定不會啟用與 的網路連線 VPCs。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 — VPC 資源的網路連線。
   - 標籤 — 您可以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 設定應用程式。如需網路連線的詳細資訊，請參閱 設定VPC存取權。
6. 若要建立應用程式，請選擇建立應用程式 。

列出應用程式

您可以從清單應用程式頁面檢視所有現有的無EMR伺服器應用程式。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。

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。應用程式應該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，您可以變更應用程式的版本標籤，將其升級至不同的 Amazon 版本EMR，也可以將架構從 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 上與您的應用程式互動 AWS CLI

從 中 AWS CLI，您可以建立、描述和刪除個別應用程式。您也可以列出所有應用程式，以便一目了然地檢視這些應用程式。本節說明如何執行這些動作。如需啟動、停止和更新應用程式等更多應用程式操作，請參閱EMR無伺服器API參考 。如需如何使用 的無EMR伺服器範例 AWS SDK for Java，請參閱我們 GitHub 儲存庫中的 Java API 範例。 https://github.com/aws-samples/emr-serverless-samples/tree/main/examples/java-api如需如何使用 的無EMR伺服器範例 AWS SDK for Python (Boto)，請參閱我們 GitHub 儲存庫中的 Python API 範例。 https://github.com/aws-samples/emr-serverless-samples/tree/main/examples/python-api

若要建立應用程式，請使用 `create-application`。您必須將 SPARK或 指定HIVE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
```

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

設定應用程式

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

主題

- 了解應用程式行為
- 預先初始化容量
- EMR Serverless 的預設應用程式組態

了解應用程式行為

預設應用程式行為

自動啟動 — 預設情況下，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 — 預設情況下，應用程式設定為在閒置 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以擴展到的最大容量。您可以指定容量上限，例如 CPU、記憶體 （GB） 和磁碟 （GB）。

> **Note**
>
> 我們建議您將最大容量設定為與支援的工作者體型成比例，方法是將工作者數量乘以其體型。例如，如果您想要將應用程式限制為 50 名工作者vCPUs，記憶體為 2、記憶體為 16 GB，磁碟為 20 GB，請將最大容量設定為 100vCPUs、記憶體為 800 GB，磁碟為 1000 GB。

支援的工作者組態

下表顯示您可以為無EMR伺服器指定的支援工作者組態和大小。您可以根據工作負載的需求，為驅動程式和執行程式設定不同的大小。

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

CPU — 每個工作者可以有 1、2、4、8 或 16 個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，這表示其使用的記憶體超過指定的容器大小。此額外負荷是以 屬性 `spark.driver.memoryOverhead`和 指定`spark.executor.memoryOverhead`。額外負荷的預設值為容器記憶體的 10%，最小值為 384 MB。當您選擇工作者規模時，應考慮此額外負荷。

例如，如果您vCPUs 為工作者執行個體選擇 4，且預先初始化的儲存容量為 30 GB，則您應該為 Spark 任務將 值設定為大約 27 GB 作為執行器記憶體。這可最大限度地提高預先初始化容量的使用率。可用的記憶體為 27 GB，加上 27 GB （ 2.7 GB ） 的 10%，總計為 29.7 GB。

磁碟：您可以使用最小大小為 20 GB 且最大大小為 200 GB 的暫存儲存磁碟來設定每個工作者。您只需為超過 20 GB 的額外儲存支付每位工作者設定的費用。

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，您可以將應用程式的 `initialCapacity` 參數設定為您要預先初始化的工作者人數。透過預先初始化的工作者容量，任務會立即開始。當您想要實作迭代應用程式和具時效性的任務時，這是理想的選擇。

當您提交任務時，如果來自 的工作者`initialCapacity`可用，該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，或者如果任務需要的資源比 提供的更多`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。當任務完成其執行時，它會釋出所使用的工作者，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並準備好使用時，應用程式才會移至 STARTED 狀態。在應用程式處於 STARTED 狀態的整個期間，EMRServerless 會保留預先初始化的容量，以供工作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 變更為 CREATED STARTED。

您可以設定應用程式在一段時間內未使用，預設為 15 分鐘時，釋出預先初始化的容量。當您提交新工作時，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，也可以在應用程式處於 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，因此您應該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，我們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 2CPUs，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 4CPUs，具有 16 GB 記憶體，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，您可以指定有多少工作者開始擔任驅動程式，以及有多少工作者開始擔任執行器。同樣地，當您使用 Apache Hive 時，您可以指定有多少工作者開始擔任 Hive 驅動程式，以及應該執行多少 Tez 任務。

設定執行具有預先初始化容量的 Apache Hive 的應用程式

下列API請求會根據 Amazon EMR版本 emr-6.6.0 建立執行 Apache Hive 的應用程式。該應用程式從 5 個預先初始化的 Hive 驅動程式開始，每個驅動程式具有 2 vCPU 和 4 GB 的記憶體，以及 50 個預先初始化的 Tez 任務工作者，每個驅動程式具有 4 vCPU 和 8 GB 的記憶體。當 Hive 查詢在此應用程式上執行時，它們會先使用已初始化的工作者，並立即開始執行。如果所有初始化前工作者都忙碌，並提交了更多 Hive 任務，則應用程式可以擴展到總共 400 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"
  }'
```

設定執行具有預先初始化容量的 Apache Spark 的應用程式

下列API請求會建立以 Amazon 6.6.0 EMR版為基礎執行 Apache Spark 3.2.0 的應用程式。該應用程式從 5 個預先初始化的 Spark 驅動程式開始，每個驅動程式具有 2 vCPU 和 4 GB 的記憶體，以及 50 個預先初始化的執行器，每個驅動程式具有 4 vCPU 和 8 GB 的記憶體。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。如果所有初始化前工作者都忙碌，並提交了更多 Spark 任務，則應用程式可以擴展到總共 400 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所需的記憶體。對於要使用預先初始化工作者的任務，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在相同應用程式下提交的所有任務指定一組通用的執行期和監控組態。這可減少與為每個任務提交相同組態需求相關聯的額外額外額外負荷。

您可以在下列時間點修改組態：

- 在任務提交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範例，以供進一步參考。

在應用程式層級宣告組態

您可以為在應用程式下提交的任務指定應用程式層級記錄和執行期組態屬性。

`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。如需記錄無EMR伺服器的詳細資訊，請參閱 儲存日誌。

`runtimeConfiguration`

若要指定執行期組態屬性，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的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無EMR伺服器版本而有所不同。例如，自訂 Log4j 的分類`spark-executor-log4j2`，`spark-driver-log4j2`且僅適用於 6.8.0 版及更新版本。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。

您也可以設定 Apache Log4j2 屬性 、 AWS Secrets Manager 以進行資料保護，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建立或更新具有秘密之無EMR伺服器應用程式的使用者和角色。

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

```
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:secretsmanager:your-secret-arn"
        }
    ]
}
```

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 的 許可政策範例 AWS Secrets Manager。

> **Note**
>
> runtimeConfiguration 您在應用程式層級指定的 會映射至 applicationConfiguration中的 StartJobRun API。

宣告範例

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.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@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 指定應用程式組態和監控組態的組態覆寫API。EMR 然後，無伺服器會合併您在應用程式層級和任務層級指定的組態，以決定任務執行的組態。

合併發生的精細程度如下：

- **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。

如需優先順序排名的詳細資訊，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請求中新增新組態參數。
- 移除現有組態 - 若要移除應用程式執行期組態 ，請提供您要移除的組態金鑰，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** - 若要移除受管持續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
            }
        }
    }'
```

執行任務時，會根據 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將新增至現有分類清單中。

- spark-driver-log4j2 將移除分類。
- 的組態managedPersistenceMonitoringConfiguration會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。

自訂無EMR伺服器映像

從 Amazon EMR 6.9.0 開始，您可以使用自訂映像，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，並讓套件更具便攜性。當您自訂無EMR伺服器映像時，它提供下列優點：

- 安裝並設定已針對工作負載最佳化的套件。這些套件可能無法廣泛用於 Amazon EMR執行期環境的公開分發。
- 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。
- 可讓您將自己的 JDK和 Python 版本用於應用程式。

EMR Serverless 提供您可以在建立自己的映像時用來作為基礎的映像。基本映像提供基本的 jar、組態和程式庫，讓映像與 Serverless EMR 互動。您可以在 Amazon ECR Public Gallery 中找到基本映像。使用符合您應用程式類型（Spark 或 Hive）和發行版本的影像。例如，如果您在 Amazon 6.9.0 EMR 版上建立應用程式，請使用下列映像。

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

必要條件

建立無EMR伺服器自訂映像之前，請先完成這些先決條件。

1. 在您 AWS 區域 用來啟動 EMR Serverless 應用程式的相同 中建立 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. 若要授予使用者對 Amazon ECR儲存庫的存取權，請將下列政策新增至使用者和角色，這些使用者和角色會使用來自此儲存庫的映像來建立或更新無EMR伺服器應用程式。

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

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，您可以包含要對映像進行的任何修改。基本映像會自動將 USER設定為 hadoop。此設定可能沒有您包含的所有修改的許可。作為解決方法，將 設定為 USER root，修改映像，然後將USER返回設定為 hadoop:hadoop。若要查看常見使用案例的範例，請參閱 搭配EMR無伺服器使用自訂映像。

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

USER root
# MODIFICATIONS GO HERE

# EMRS will run the image as hadoop
USER hadoop:hadoop
```

取得 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。如需有關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
```

您應該會看到類似下列的輸出。

```
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 Amazon ECR映像推送至 Amazon ECR儲存庫。確保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 Management Console 索引標籤或 AWS CLI 索引標籤，然後完成下列步驟。

Console

1. 在 https://console.aws.amazon.com/emr 登入 EMR Studio 主控台。導覽至您的應用程式，或使用建立應用程式 中的說明建立新的應用程式。
2. 若要在建立或更新無EMR伺服器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，選取將自訂映像與此應用程式搭配使用核取方塊。
4. 將 Amazon ECR映像貼URI到映像URI欄位中。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URIs為每個工作者類型貼上不同的Amazon ECR映像。

CLI

- 使用 `image-configuration` 參數建立應用程式。EMR Serverless 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 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 `describe`、 `get`和 `download`請求。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Emr Serverless Custom Image Support",
      "Effect": "Allow",
      "Principal": {
        "Service": "emr-serverless.amazonaws.com"
      },
      "Action": [
        "ecr:BatchGetImage",
        "ecr:DescribeImages",
        "ecr:GetDownloadUrlForLayer"
      ],
      "Condition":{
        "StringEquals":{
          "aws:SourceArn": "arn:aws:emr-serverless:region:aws-account-id:/
applications/application-id"
        }
      }
    }
  ]
}
```

作為安全最佳實務，將`aws:SourceArn`條件索引鍵新增至儲存庫政策。IAM 全域條件金鑰`aws:SourceArn`可確保 EMR Serverless 僅將儲存庫用於應用程式 ARN。如需 Amazon ECR儲存庫政策的詳細資訊，請參閱建立私有儲存庫。

考量與限制

當您使用自訂映像時，請考慮下列事項：

- 使用與應用程式類型（ Spark 或 Hive ）和版本標籤（ 例如 `emr-6.9.0` ）相符的正確基本映像。
- EMR Docker 檔案中的無伺服器忽略`[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，您不應修改環境變數 `JAVA_HOME`、`SPARK_HOME`、`HIVE_HOME`、 。

- 自訂映像的大小不得超過 5 GB。
- 如果您在 Amazon EMR基礎映像中修改二進位檔或 jar，可能會導致應用程式或任務啟動失敗。
- Amazon ECR儲存庫應與 AWS 區域 啟動 EMR Serverless 應用程式時所使用的儲存庫相同。

設定VPC存取權

您可以設定無EMR伺服器應用程式來連線至 內的資料存放區VPC，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 應用程式具有傳出連線，可連線至您 內的資料存放區VPC。根據預設，無EMR伺服器會封鎖對應用程式的傳入存取，以改善安全性。

> **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存取權。如需有關如何設定外部 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，您可以選擇自訂設定，並指定無EMR伺服器應用程式可以使用的 VPC、子網路和安全群組。

VPCs

選擇包含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VPCs所選 的所有 AWS 區域。

子網

選擇 中VPC包含資料存放區的子網路。建立應用程式頁面會列出 中資料存放區的所有子網路VPC。

選取的子網路必須是私有子網路。這表示子網路的相關路由表不應具有網際網路閘道。

對於網際網路的傳出連線，子網路必須使用NAT閘道具有傳出路由。若要設定NAT閘道，請參閱使用NAT閘道 。

對於 Amazon S3 連線，子網路必須設定NAT閘道或VPC端點。若要設定 S3 VPC端點，請參閱建立閘道端點 。

若要連線至 AWS 服務 以外的其他 VPC，例如 Amazon DynamoDB ，您必須設定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 。

工作者可以透過VPC傳出流量連線到 內的資料存放區。根據預設，無EMR伺服器會封鎖對工作者的傳入存取，以改善安全性。

當您使用 時 AWS Config，EMRServerless 會為每個工作者建立彈性網路介面項目記錄。若要避免與此資源相關的成本，請考慮在 `AWS::EC2::NetworkInterface`中關閉 AWS Config。

> **Note**
>
> 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。每個工作者都會在其啟動的子網路上消耗 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”。

安全群組

選擇一個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 中的所有安全群組 VPC。EMR 無伺服器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**
>
> 建議您為無EMR伺服器應用程式建立個別的安全群組。這可讓隔離和管理網路規則更有效率。例如，若要與 Amazon Redshift 叢集通訊，您可以定義 Redshift 和無EMR伺服器安全群組之間的流量規則，如以下範例所示。

Example 範例 — 與 Amazon Redshift 叢集通訊

1. 從其中一個無EMR伺服器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
全部 TCP	TCP	5439	`emr-serverless-security-group`

2. 新增來自其中一個無EMR伺服器安全群組的傳出流量規則。您可以使用兩種方式的其中一種來執行此動作。首先，您可以開啟所有連接埠的傳出流量。

Type	通訊協定	連接埠範圍	目的地
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，而且沒有其他功能時，此功能才有用。

Type	通訊協定	連接埠範圍	來源
全部 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無EMR伺服器應用程式的網路組態。

檢視任務執行詳細資訊

在任務執行詳細資訊頁面上，您可以檢視任務用於特定執行的子網路。請注意，任務僅在從指定子網路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 中可用 IP 地址的子集VPC。例如，VPC具有 /16 網路遮罩的 最多有 65，536 個可用的 IP 地址，可以使用子網路遮罩分解為多個較小的網路。例如，您可以使用 /17 遮罩和 32，768 個可用的 IP 地址，將此範圍分割為兩個子網路。子網路位於可用區域內，無法跨區域。

子網路的設計應記住您的 EMR Serverless 應用程式擴展限制。例如，如果您的應用程式請求 4 vCpu名工作者，並且可以擴展到 4，000 vCpu名，則您的應用程式最多需要 1，000 名工作者，總共需要 1，000 個網路介面。建議您跨多個可用區域建立子網路。這可讓 EMR Serverless 在極少數情況下，當可用區域故障時，重試您的任務或將預先初始化的容量佈建到不同的可用區域中。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 個 IP 地址，而 /21 的遮罩提供 2048 個 IP 地址。以下是 /16 網路遮罩中 4 個子網路的範例，其中 /22 VPC 遮罩可以配置到不同的可用區域。可用 IP 地址與可用 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 預留 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 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	10.0.4.0 - 10.0.7.255	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最適合較大的工作者規模。使用較大的工作者規模需要的網路介面較少。例如，使用 16 vCpu 名具有 4，000 個應用程式擴展限制的工作者 vCpu ，最多需要 250 名工作者，總共需要 250 個可用的 IP 地址來佈建網路介面。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。

如果您在多個應用程式之間共用子網路，則每個子網路的設計都應牢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 vCpu 名工作者，且每個應用程式可以擴展至 4000 vCpu 個，並以 12，000 個vCpu帳戶層級服務為基礎的配額為基礎，則每個子網路都需要 3000 個可用的 IP 地址。如果您想要VPC使用的 IP 地址數量不足，請嘗試增加可用的 IP 地址數量。您可以建立其他無類別網域間路由 （CIDR） 區塊與 的關聯，藉此實現此目標VPC。如需詳細資訊，請參閱 Amazon VPC使用者指南 中的將其他IPv4CIDR區塊與 建立關聯VPC。

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決定應用程式用來執行任務的處理器類型。Amazon 為您的應用程式EMR提供兩種架構選項：x86_64 和 arm64。EMR 無伺服器會在最新一代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

x86_64 架構也稱為 x86 64 位元或 x64。x86_64 是無EMR伺服器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 位元相容ARM，則您可以切換至 arm64，以使用 Graviton 處理器來改善效能、運算能力和記憶體。與在 x86 架構上執行大小相等的執行個體相比，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 上可用的 AWS 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。EMR 無伺服器會在可用時自動使用最新一代的處理器，而不需要您為升級至最新一代的處理器而付出任何努力。

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 中指定 `create-application` `ARM64`作為 `architecture` 參數API。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

您可以設定現有的 Amazon EMR Serverless 應用程式，將 Graviton （ arm64 ） 架構與 SDK AWS CLI、 或 EMR Studio 搭配使用。

若要將現有應用程式從 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 \
 --application-id application-id \
 --region us-west-2
```

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

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

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

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

5. 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

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 ） 作為架構選項時，請確定第三方套件和程式庫與 64 位元ARM架構相容。如需如何將 Python 程式庫封裝至與所選架構相容的 Python 虛擬環境中的詳細資訊，請參閱 搭配EMR無伺服器使用 Python 程式庫。

若要進一步了解如何設定 Spark 或 Hive 工作負載以使用 64 位元 ARM，請參閱 上的 AWS Graviton 入門儲存庫 GitHub。此儲存庫包含可協助您開始使用 ARM型 Graviton 的重要資源。

任務並行和佇列

從 Amazon 7.0.0 版及更新EMR版本開始，您可以為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 會從佇列您的任務開始，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

當需要提交許多任務時，任務並行和佇列提供下列優點：

- 它有助於控制並行執行任務，以有效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

下列程序顯示實作並行和佇列的幾種不同方法。

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。若要這麼做，應用程式必須具有版本標籤 emr-7.0.0 或更新版本。

使用 AWS Management Console

下列步驟說明如何使用 開始使用任務並行和佇列 AWS Management Console：

1. 前往 EMR Studio 並選擇建立版本標籤為 EMR-7.0.0 或更高版本的應用程式。
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 功能。
4. 選取後，您可以同時選取任務執行和佇列逾時，以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，然後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，並檢查屬性索引標籤下的 ，以判斷功能是否已啟用。

在組態之後，您可以在啟用此功能的情況下提交任務。

並行和佇列的考量

當您實作並行和佇列時，請考慮下列事項：

- Amazon 7.0.0 版及更新EMR版本支援任務佇列和並行。
- 您可以為 STARTED 狀態中的應用程式更新並行。
- 的有效範圍`maxConcurrentRuns`為 1 到 1000，`queueTimeoutMinutes`而 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無伺服器將資料匯入 S3 快速單一區域

在 Amazon 7.2.0 及更高EMR版本中，您可以將EMR無伺服器與 Amazon S3 Express 單區儲存類別搭配使用，以提高執行任務和工作負載時的效能。S3 Express One Zone 是一種高效能的單區域 Amazon S3 儲存類別，可為大多數對延遲敏感的應用程式提供一致、10 毫秒的資料存取。在發布時，S3 Express One Zone 提供 Amazon S3 中最低延遲和最高效能的雲端物件儲存。

必要條件

- S3 快速單一區域許可 — 當 S3 Express 單一區域初始執行動作 (例如 `GETLIST`，或PUT在 S3 物件上) 時，儲存類別會代表您呼叫`CreateSession`。您的IAM政策必須允許`s3express:CreateSession`權限，以便 S3A 連接器可以呼叫 `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 S3 快速單區儲存類別的 Amazon S3 儲存貯體存取資料，您必須使用 Apache Hadoop 連接器 S3A。 若要使用連接器，請確保所有 S3 都URIs使用`s3a`配置。如果沒有，您可以變更用於 `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

請按照下列步驟開始使用 S3 快速單區。

1. 建立VPC端點。將端點新增 `com.amazonaws.us-west-2.s3express`到VPC端點。
2. 請遵循 Amazon EMR 無伺服器入門操作，建立具有 Amazon EMR 版本標籤 7.2.0 或更高版本的應用程式。
3. 將您的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`權限新增至您的工作執行角色。

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

5. 執行您的工作。請注意，您必須使用S3A配置來存取 S3 Express 單一區域儲存貯體。

```
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

佈建應用程式之後，您可以將工作提交至應用程式。本節介紹如何使用 AWS CLI 以執行這些工作。本節也會識別EMR無伺服器上可用之每種應用程式類型的預設值。

主題

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至 ，RUNNING直到達到 FAILED、 SUCCESS或 CANCELLING為止。

作業執行可能有以下狀態：

州	描述
已提交	當您將任務執行提交至 EMR Serverless 時的初始任務狀態。任務正在等待為應用程式排程。EMR Serverless 開始排定任務執行的優先順序和排程。

州	描述
已佇列	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 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 已排程應用程式的任務執行，並正在配置資源來執行任務。
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，或任務執行失敗。如需此任務失敗的其他資訊`StateDetails` ，請參閱 。
Success	任務執行已成功完成。
取消	`CancelJobRun` 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。

從 EMR Studio 主控台執行工作

您可以將工作執行提交至EMR無伺服器應用程式，並從 EMR Studio 主控台檢視工作。若要在 EMR Studio 主控台上建立或瀏覽至EMR無伺服器應用程式，請遵循從主控台開始使用中的指示。

提交工作

在 [送出工作] 頁面上，您可以依照下列步驟將工作送出至EMR無伺服器應用程式。

Spark

1. 在「名稱」欄位中，輸入工作執行的名稱。
2. 在「執行時間角色」欄位中，輸入EMR無伺服器應用程式在工作執行時可承擔的IAM角色名稱。若要進一步瞭解執行階段角色，請參閱Amazon EMR Serverless 的任務執行期角色。
3. 在指令碼位置欄位中，輸入指令碼或您要執行JAR的 Amazon S3 位置。對於星火作業，腳本可以是一個 Python（ `.py` ）文件或JAR（ `.jar` ）文件。
4. 如果您的指令集位置是JAR檔案，請在「主要類別」欄位中輸入作為工作進入點的類別名稱。
5. (選擇性) 輸入其餘欄位的值。
   - 指令碼引數 — 輸入要傳遞至主要JAR或 Python 指令集的任何引數。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。
   - 星火屬性 — 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 驅動程序和執行程序大小，則必須考慮內存開銷。在屬性`spark.driver.memoryOverhead`和中指定記憶體額外負荷值`spark.executor.memoryOverhead`。記憶體額外負荷的預設值為 10% 的容器記憶體，最少為 384 MB。執行程序內存和內存開銷一起不能超過工作內存。例如，30 GB 工作`spark.executor.memory`站的最大值必須是 27 GB。

   - Job 組態 — 在此欄位中指定任何工作組態。您可以使用這些工作組態來覆寫應用程式的預設組態。
   - 其他設置-激活或停用 AWS Glue 資料型錄做為中繼儲存庫，並修改應用程式記錄設定。若要進一步瞭解中繼儲存區設定，請參閱Metastore 組態。若要進一步瞭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，輸入工作執行的名稱。
2. 在「執行時間角色」欄位中，輸入EMR無伺服器應用程式在工作執行時可承擔的IAM角色名稱。

3. 在指令碼位置欄位中，輸入指令碼或您要執行JAR的 Amazon S3 位置。對於 Hive 作業，指令碼必須是 Hive (`.sql`) 檔案。
4. (選擇性) 輸入其餘欄位的值。

   - 初始化指令碼位置 — 輸入 Hive 命令檔執行之前，初始化表格的指令碼位置。
   - 配置單元屬性-展開蜂房屬性部分，並在此字段中輸入任何 Hive 配置參數。
   - Job 組態 — 指定任何工作組態。您可以使用這些工作組態來覆寫應用程式的預設組態。對於 Hive 作業，`hive.exec.scratchdir`並且`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 資料型錄做為中繼儲存庫，並修改應用程式記錄設定。若要進一步瞭解中繼儲存區設定，請參閱Metastore 組態。若要進一步瞭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)。

檢視任務執行

您可以從應用程式「詳細資訊」頁面上的「J ob 執行」標籤檢視工作執行，並針對工作執行執行執行執行執行以下動作。

取消工作 — 若要取消處於此RUNNING狀態的工作執行，請選擇此選項。若要進一步瞭解工作執行轉換，請參閱作業執行狀態。

複製工作 — 若要複製先前執行的工作並重新提交，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，以一目了然地檢視這些任務。

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 Spark 任務。如需 Hive 範例，請參閱 Hive 任務。此命令會傳回您的 ARN、`application-id`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，EMRServerless 會自動取消它。預設逾時為 12 小時。開始任務執行時，您可以將此逾時設定設定為符合您任務要求的值。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。如果您不想查看所有任務，您可以指定要查看的任務數量上限，最多 50 個。下列範例指定您要查看兩個上次任務執行。

```
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無伺服器API參考 。

使用隨機最佳化磁碟

在 Amazon 7.1.0 及更高EMR版本中，您可以在執行 Apache Spark 或 Hive 任務時使用隨機最佳化的磁碟，以改善 I/O 密集型工作負載的效能。與標準磁碟相比，隨機播放最佳化磁碟可提供更高 IOPS (每秒 I/O 作業數)，以加快資料移動速度並減少隨機播放作業期間的延遲。隨機最佳化磁碟可讓您連接每個 Worker 最多 2 TB 的磁碟大小，因此您可以根據工作負載需求設定適當的容量。

主要優點

隨機最佳化磁碟具有下列優點。

- 高IOPS效能 — 隨機最佳化磁碟可提供IOPS比標準磁碟更高的磁碟，在 Spark 和 Hive 任務以及其他隨機執行密集型工作負載期間，進行資料清洗效率更快速。
- 更大的磁碟大小 — 隨機最佳化磁碟支援每位工作者 20GB 到 2TB 的磁碟大小，因此您可以根據工作負載選擇適當的容量。

開始使用

請參閱下列步驟，以便在工作流程中使用隨機播放最佳化磁碟。

Spark

1. 使用下列EMR命令建立無伺服器 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. 將 Spark 工作設定為包含參數spark.emr-serverless.driver.disk.type和/或spark.emr-serverless.executor.disk.type使用隨機最佳化磁碟執行。您可以使用一個或兩個參數，具體取決於您的使用案例。

```
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 工作屬性。

Hive

1. 使用下列EMR命令建立無伺服器 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. 將 Hive 工作設定為包含參數hive.driver.disk.type和/或hive.tez.disk.type使用隨機最佳化磁碟執行。您可以使用一個或兩個參數，具體取決於您的使用案例。

```
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，Hive 工作屬性。

使用預先初始化容量設定應用程式

請參閱下列範例，以建立以 Amazon 7.1.0 EMR 版為基礎的應用程式。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 v CPU、4 GB 的記憶體，以及 50 GB 的隨機播放最佳化磁碟。

- 50 個預先初始化的執行程序，每個執行程序都具有 4 vCPU，8 GB 的內存和 500 GB 的隨機播放優化磁盤。

當此應用程式執行 Spark 作業時，它會先使用預先初始化的 Worker，然後將隨選背景工作程式擴充到最大容量 400 v CPU 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"
}'
```

串流工作

EMR無伺服器中的串流工作是一種工作模式，可讓您以近乎即時的速度分析和處理串流資料。這些長時間執行的工作輪詢串流資料，並在資料到達時持續處理結果。串流作業最適合需要即時資料處理的工作，例如近乎即時的分析、詐騙偵測和建議引擎。EMR無伺服器串流作業可提供最佳化功能，例如內建工作備援、即時監控、增強的記錄管理，以及與串流連接器整合。

以下是一些串流工作的使用案例：

- 近乎即時的分析 — Amazon EMR Serverless 中的串流任務可讓您以近乎即時的方式處理串流資料，因此您可以對連續資料串流 (例如日誌資料、感應器資料或點擊流資料) 執行即時分析，以獲得見解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發生時，您可以使用串流作業在金融交易、信用卡作業或線上活動中執行近乎即時的詐騙偵測。

- 建議引擎 — 串流作業可以處理使用者活動資料並更新建議模型。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 — 串流工作可以處理社交媒體資料，例如推文、留言和貼文，因此組織可以近乎即時地監控趨勢、情緒分析和管理品牌聲譽。
- 物聯網 (IoT) 分析 — 串流任務可以處理和分析來自 IoT 裝置、感應器和連線機械的高速資料串流，因此您可以執行異常偵測、預測性維護和其他 IoT 分析使用案例。
- 點擊流分析 — 流任務可以處理和分析來自網站或移動應用程序的點擊流數據。使用此類資料的企業可以執行分析，進一步瞭解使用者行為、個人化使用者體驗，以及最佳化行銷宣傳活動。
- 記錄監控和分析 — 串流作業也可以處理來自伺服器、應用程式和網路裝置的記錄資料。這可為您提供異常偵測、疑難排解，以及系統健康狀態和效能。

主要優點

EMR無伺服器中的串流作業會自動提供工作備援，這是下列因素的組合：

- 自動重試 — EMR 無伺服器會自動重試任何失敗的工作，而無需您手動輸入。
- 可用區域 (AZ) 備援 — 如果原始 AZ 發生問題，EMR無伺服器會自動將串流作業切換至運作良好的可用區域。
- 日誌管理：
  - 記錄輪替 — 為了更有效率的磁碟儲存管理，EMRServerless 會定期輪換長串流工作的記錄檔。這樣做可防止記錄累積可能會消耗所有磁碟空間。
  - 記錄壓縮 — 協助您有效率地管理及最佳化受管理持續性中的記錄檔。當您使用受管理的 spark 歷程記錄伺服器時，壓縮也會改善偵錯體驗。

支援的資料來源和資料接收器

EMR無伺服器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、適用於 Apache 卡夫卡的 Amazon 受管串流，以及自我管理的 Apache 卡夫卡叢集。根據預設，Amazon 7.1.0 及更高EMR版本包含 Amazon Kinesis Data Streams 連接器，因此您不需要建立或下載任何額外的套件。
- 支援的輸出資料接收器 — AWS Glue 數據目錄表，Amazon S3，Amazon Redshift，我的，Postgre SQL 甲骨文SQL，甲骨文，Microsoft，阿帕奇冰山SQL，三角洲湖和阿帕奇胡迪。

考量與限制

使用串流工作時，請記住下列考量事項和限制。

- Amazon 7.1.0 及更高EMR版本支援串流任務。
- EMRServerless 預期串流作業會長時間執行，因此您無法設定執行逾時來限制工作的執行時間。
- 串流作業僅與 Spark 引擎相容，該引擎建立在結構化串流架構之上。
- EMR無伺服器會無限期地重試串流作業，而且您無法自訂最大嘗試次數。如果失敗的嘗試次數超過了每小時窗口設置的閾值，則會自動包含鞭打防止以停止工作重試。預設臨界值為一小時內的五次失敗嘗試。您可以將此臨界值設定為 1 到 10 次嘗試之間。如需詳細資訊，請參閱 Job 復原。
- 串流工作具有儲存執行階段狀態和進度的檢查點，因此EMR無伺服器可以從最新的檢查點繼續串流工作。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。

開始串流工作

請參閱下列指示，瞭解如何開始使用串流工作。

1. 請遵循開始使用 Amazon EMR 無伺服器建立應用程式。請注意，您的應用程式必須執行 Amazon 7.1.0 或更高EMR版本。
2. 應用程式準備就緒後，請將mode參數設定STREAMING為提交串流工作，類似下列內容 AWS CLI 例子。

```
aws emr-serverless start-job-run \
--application-id <APPPLICATION_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 連接器可讓您建立串流應用程式和管道，以取用來自 Amazon Kinesis 資料串流的資料並將資料寫入。此連接器支援增強的扇出消耗，每個碎片的專用讀取輸送率最高可達 2MB/ 秒。根據預設，Amazon EMR 無伺服器 7.1.0 及更高版本包含連接器，因此您不需要建立或下載任何額外的套件。如需有關連接器的詳細資訊，請參閱上的 spark-sql-kinesis-connector 頁面 GitHub。

以下是如何使用 Kinesis Data Streams 連接器相依性開始工作執行的範例。

```
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"
    }
}'
```

若要連線到 Kinesis Data Streams，您必須設定具有VPC存取權的EMR無伺服器應用程式，並使用VPC端點來允許私人存取。或使用NAT閘道取得公開存取權。如需詳細資訊，請參閱設定VPC存取權。您也必須確定您的工作執行階段角色具有必要的讀取和寫入權限，才能存取所需的資料串流。若要進一步了解如何設定 Job 務執行時期角色，請參閱 Amazon EMR 無伺服器的任務執行階段角色。如需所有必要權限的完整清單，請參閱上的spark-sql-kinesis-connector 頁面 GitHub。

阿帕奇卡夫卡連接器

用於星火結構化流阿帕奇卡夫卡連接器是來自星火社區的開源連接器，並在 Maven 存儲庫中可用。此連接器可協助 Spark 結構化串流應用程式讀取資料，並將資料寫入自我管理的 Apache Kafka，以及適用於 Apache Kafka 的 Amazon 受管串流。如需有關連接器的詳細資訊，請參閱 Apache Spark 說明文件中的結構化串流 + 卡夫卡整合指南。

下列範例示範如何在工作執行要求中包含 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無伺服器發行版本和對應的 Spark 版本。要查找正確的卡夫卡版本，請參閱結構化流 + 卡夫卡集成指南。

若要透過IAM身份驗證使用適用於 Apache Kafka 的 Amazon 受管串流，您必須包含另一個相依性，以使 Kafka 連接器能夠透過連接到 Amazon。MSK IAM如需詳細資訊，請參閱（詳見）的aws-msk-iam-auth 存放庫 GitHub。您也必須確定工作執行階段角色具有必要的IAM權限。下列範例示範如何將連接器與IAM驗證搭配使用。

```
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>"
    }
}'
```

若要使用來自 Amazon 的 Kafka 連接器和IAM身份驗證程式庫，MSK您必須設定具有存取權的EMR無伺服器應用程式。VPC您的子網必須具有 Internet 訪問權限，並使用NAT網關訪問 Maven 依賴關係。如需詳細資訊，請參閱設定VPC存取權。子網路必須具有網路連線能力才能存取 Kafka 叢集。無論您的 Kafka 叢集是自我管理還是使用適用於 Apache Kafka 的 Amazon 受管串流，都是如此。

串流工作記錄管理

日誌旋轉

串流工作支援 Spark 應用程式記錄檔和事件記錄檔的記錄輪替。記錄輪換可防止長時間串流工作產生可能會佔用所有可用磁碟空間的大型記錄檔。記錄輪替可協助您節省磁碟儲存空間，並防止因磁碟空間不足而導致工作失敗。如需詳細資訊，請參閱輪替記錄檔。

日誌壓實

只要有受管理的記錄檔可用，串流作業也支援 Spark 事件記錄的記錄壓縮。如需受管理記錄的詳細資訊，請參閱使用受管理儲存區記錄。串流作業可能會長時間執行，而且隨著時間的推移，事件資料量可能會大幅增加記錄檔大小。星火歷史記錄服務器讀取並將這些事件加載到內存中的 Spark 應用程序 UI。此程序可能會導致高延遲和成本，尤其是存放在 Amazon S3 中的事件日誌非常大時。

記錄壓縮會減少事件記錄檔的大小，因此 Spark 歷程記錄伺服器不需要載入超過 1 GB 的事件記錄檔在任何時間。如需詳細資訊，請參閱 Apache Spark 文件中的監視和檢測。

Spark 任務

您可以在 `type` 參數設定為 的應用程式上執行 Spark 任務SPARK。任務必須與與 Amazon EMR發行版本相容的 Spark 版本相容。例如，當您使用 Amazon 6.6.0 EMR版執行任務時，您的任務必須與 Apache Spark 3.2.0 相容。如需每個版本應用程式版本的資訊，請參閱 Amazon EMR 無伺服器發行版本。

Spark 任務參數

當您使用 `StartJobRun API` 執行 Spark 任務時，您可以指定下列參數。

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

- 從 S3 儲存貯體或資料所在的其他資料來源讀取
- 從 PySpark 指令碼或JAR檔案所在的 S3 儲存貯體或字首讀取
- 寫入您打算寫入最終輸出的 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 Spark，存取 AWS Glue Data CatalogSQL

如果您的 Spark 任務讀取或寫入來自其他資料來源的資料，請在此IAM角色中指定適當的許可。如果您未提供這些許可給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

`jobDriver` 使用 為任務提供輸入。任務驅動程式參數只接受一個您要執行的任務類型值。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、PySparkSparkR 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`引數中定義的執行器數量。

- **entryPointArguments** – 這是您要傳遞至主檔案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 中您要執行的主檔案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 中指定主要項目類別。

如需其他資訊，請參閱透過 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** - 使用此欄位來指定 Amazon S3 URL（`s3MonitoringConfiguration`），您希望 EMR Serverless 任務在其中儲存 Spark 任務的日誌。請確定您已使用與託管應用程式相同的 AWS 帳戶 建立此儲存貯體，並在執行任務 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。您可以使用短期語法來提供組態，也可以參考JSON檔案中的組態物件。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。您可以為單一JSON物件中的多個應用程式指定多個分類。

> **Note**
>
> 可用的組態分類會因特定無EMR伺服器版本而有所不同。例如，自訂 Log4j 的分類`spark-executor-log4j2`，`spark-driver-log4j2`且僅適用於 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 提交參數會優先處理。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`。
- 您以 提供的組態`sparkSubmitParameters`與 `--conf` 引數。
- 當您啟動任務時，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 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，以更好地調整 Spark 請求和取消執行器的速度與 EMR Serverless 建立和釋放工作者的速度。如此一來，EMRServerless 就更能在各個階段以最佳方式重複使用工作者，因此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：

- 此最佳化適用於您為其啟用動態資源配置的 Spark 任務。
- 為了獲得最佳成本效率，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的上擴展範圍。
- 您可能無法看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，或在單一階段完成執行，Spark 可能不需要更多執行器或多個擴展事件。
- 具有大量階段、較小階段，然後再次出現大型階段的任務，可能會在任務執行期中經歷迴歸。隨著 EMR Serverless 更有效率地使用資源，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。

金鑰	描述	預設值
spark.archives	Spark 擷取到每個執行器工作目錄的封存逗號分隔清單。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，請在要擷取的檔案名稱#後面新增 。例如：file.zip#directory 。	NULL
spark.authenticate	開啟 Spark 內部連線身分驗證的選項。	TRUE
spark.driver.cores	驅動程式使用的核心數量。	4
spark.driver.extraJavaOptions	Spark 驅動程式的其他 Java 選項。	NULL
spark.driver.memory	驅動程式使用的記憶體量。	14G
spark.dynamicAllocation.enabled	開啟動態資源配置的選項。此選項會根據工作負載，將向應用程式註冊的執行程式數量向上或向下擴展。	TRUE
spark.dynamicAllocation.executorIdleTimeout	Spark 移除執行器之前，執行器可以保持閒置的時間長度。這僅適用於開啟動態配置的情況。	60 秒
spark.dynamicAllocation.initialExecutors	如果您開啟動態配置，要執行的執行器初始數量。	3
spark.dynamicAllocation.maxExecutors	如果您開啟動態配置，則執行器數量的上限。	對於 6.10.0 和更新版本，infinity

金鑰	描述	預設值
		對於 6.9.0 和更低版本， 100
spark.dynamicAllocation.minExecutors	如果您開啟動態配置，執行器數量的下限。	0
spark.emr-serverless.allocation.batch.size	執行器配置的每個週期中要請求的容器數量。每個配置週期之間有一秒的差距。	20
spark.emr-serverless.driver.disk	Spark 驅動程式磁碟。	20G
spark.emr-serverless.driverEnv. *[KEY]*	將環境變數新增至 Spark 驅動程式的選項。	NULL
spark.emr-serverless.executor.disk	Spark 執行器磁碟。	20G
spark.emr-serverless.memoryOverheadFactor	設定記憶體額外負荷，以新增至驅動程式和執行器容器記憶體。	0.1
spark.emr-serverless.driver.disk.type	連接至 Spark 驅動程式的磁碟類型。	標準
spark.emr-serverless.executor.disk.type	連接至 Spark 執行器的磁碟類型。	標準
spark.executor.cores	每個執行器使用的核心數目。	4
spark.executor.extraJavaOptions	Spark 執行器的其他 Java 選項。	NULL
spark.executor.instances	要配置的 Spark 執行器容器數量。	3

金鑰	描述	預設值
spark.executor.memory	每個執行器使用的記憶體量。	14G
spark.executorEnv.*[KEY]*	將環境變數新增至 Spark 執行器的選項。	NULL
spark.files	每個執行器的工作目錄中要進入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SparkFiles.get(*fileName*)。	NULL
spark.hadoop.hive.metastore.client.factory.class	Hive 中繼存放區實作類別。	NULL
spark.jars	要新增至驅動程式和執行器執行期類別路徑的其他 jar。	NULL
spark.network.crypto.enabled	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
spark.sql.warehouse.dir	受管資料庫和資料表的預設位置。	的值 $PWD/spark-warehouse
spark.submit.pyFiles	以逗號分隔的清單.zip，列出要放置在 PYTHONPATH Python 應用程式中的 .egg、或 .py 檔案。	NULL

下表列出預設 Spark 提交參數。

金鑰	描述	預設值
archives	Spark 擷取到每個執行器工作目錄的封存逗號分隔清單。	NULL
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目。	4
executor-memory	執行器使用的記憶體量。	14G
files	以逗號分隔的檔案清單，以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SparkFiles.get(*fileName*)。	NULL
jars	要包含在驅動程式和執行器類別路徑上的以逗號分隔的 jar 清單。	NULL
num-executors	要啟動的執行程式數目。	3
py-files	以逗號分隔的清單.zip，列出要放置在PYTHONPATH 適用於 Python 應用程式的 上的.egg、 或 .py 檔案。	NULL
verbose	開啟其他偵錯輸出的選項。	NULL

Spark 範例

下列範例示範如何使用 StartJobRunAPI來執行 Python 指令碼。如需使用此範例的教學課程 end-to-end，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 StartJobRunAPI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"
        }
    }'
```

Hive 任務

您可以在 type 參數設定為 的應用程式上執行 Hive 任務HIVE。任務必須與與 Amazon EMR發行版本相容的 Hive 版本相容。例如，當您在具有 Amazon 6.6.0 EMR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。如需每個版本應用程式版本的資訊，請參閱 Amazon EMR 無伺服器發行版本。

Hive 任務參數

當您使用 `StartJobRun API` 執行 Hive 任務時，您必須指定下列參數。

必要參數

- Hive 任務執行期角色
- Hive 任務驅動程式參數
- Hive 組態覆寫參數

Hive 任務執行期角色

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

- 從 S3 儲存貯體或資料所在的其他資料來源讀取
- 從 Hive 查詢檔案和初始查詢檔案所在的 S3 儲存貯體或字首讀取
- 讀取和寫入您的 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 任務讀取或寫入來自其他資料來源的資料，請指定此IAM角色的適當許可。如果您未提供這些許可給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

`jobDriver` 使用 為任務提供輸入。任務驅動程式參數僅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，EMRServerless 會將 Hive 查詢傳遞至 `jobDriver` 參數。Hive 任務具有下列參數：

- **`query`** – 這是 Amazon S3 中您要執行的 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 的形式傳遞至此參數`--hiveconf property=value`。若要覆寫變數，請以 的形式將其傳遞至此參數`--hivevar key=value`。
- **initQueryFile** – 這是 init Hive 查詢檔案。Hive 會在查詢之前執行此檔案，並且可以使用它來初始化資料表。

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** – 使用此欄位來指定 Amazon S3 URL（`s3MonitoringConfiguration`），您希望 EMR Serverless 任務儲存 Hive 任務日誌的位置。請確定您建立此儲存貯體時，與託管應用程式 AWS 帳戶 相同，且位於執行任務 AWS 區域 的相同位置。
- **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。您可以使用短期語法來提供組態，也可以參考JSON檔案中的組態物件。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。您可以為單一JSON物件中的多個應用程式指定多個分類。

> **Note**
>
> 可用的組態分類會因特定無EMR伺服器版本而有所不同。例如，自訂 Log4j 的分類`spark-driver-log4j2`、`spark-executor-log4j2`，且僅適用於 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入最低優先順序。

- 您透過 作為 Hive 參數的一部分提供的組態`--hiveconf property=value`。
- 當您啟動任務時，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 為版本EMR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

下表列出您在提交 Hive 任務時必須設定的強制性屬性。

設定	描述
hive.exec.scratchdir	在 Hive 任務執行期間，EMRServerless 建立暫存檔案的 Amazon S3 位置。
hive.metastore.warehouse.dir	Hive 中受管資料表資料庫的 Amazon S3 位置。

下表列出選用的 Hive 屬性，以及提交 Hive 任務時可以覆寫的預設值。

設定	描述	預設值
fs.s3.customAWSCredentialsProvider	您要使用的 AWS 憑證提供者。	com.amazonaws.auth.DefaultAWSCredentialsProviderChain
fs.s3a.aws.credentials.provider	您要搭配 S3A 檔案系統使用的 AWS 憑證提供者。	com.amazonaws.auth.DefaultAWSCredentialsProviderChain
hive.auto.convert.join	根據輸入檔案大小，開啟常見聯結自動轉換為映射聯結的選項。	TRUE
hive.auto.convert.join.noconditionaltask	當 Hive 根據輸入檔案大小將常見聯結轉換為映射聯結時，開啟最佳化的選項。	TRUE
hive.auto.convert.join.noconditionaltask.size	聯結會直接轉換為低於此大小的映射聯結。	根據 Tez 任務記憶體計算最佳值

設定	描述	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。設定為時false，Tez AM 會在 Hive 查詢編譯之後啟動。	TRUE
hive.compute.query.using.stats	啟用 Hive 以使用存放於中繼存放區中的統計資料來回答特定查詢的選項。對於基本統計資料，請將 hive.stats.autogather 設為 TRUE。如需更進階的查詢集合，請執行 analyze table queries。	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目。	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

設定	描述	預設值
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 會與 20% 的頂部空間平均共用此記憶體。	6G
hive.emr-serverless.launch.env.[*KEY*]	在所有 Hive 特定程序中設定*KEY*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/ 中開啟動態分割區的選項DDL。	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，您必須指定至少一個靜態分割區，以防意外覆寫所有分割區。在非嚴格模式下，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.pernode	Hive 在每個映射器和減少器節點中建立的動態分割區數量上限。	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 指定您要花較少的時間在分割產生中，而不是查詢執行。 ETL 指定您要花更多的時間在分割產生中。HYBRID 指定根據啟發式選擇上述策略。	HYBRID

設定	描述	預設值
hive.exec.reducers.bytes.per.reducer	每個精簡器的大小。預設值為 256 MB。如果輸入大小為 1G 任務會使用 4 個減少程式。	256000000
hive.exec.reducers.max	減少程式數目上限。	256
hive.exec.stagingdir	存放 Hive 在資料表位置和 hive.exec.scratchdir 屬性中指定的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 可以將選取的查詢轉換為單一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	開啟精簡器投機啟動的選項。僅支援 Amazon EMR 6.10.x 及更低版本。	TRUE

設定	描述	預設值
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
hive.merge.mapfiles	選項，導致小型檔案在僅映射任務結束時合併。	TRUE
hive.merge.size.per.task	任務結束時合併檔案的大小。	256000000
hive.merge.tezfiles	在 Tez 結束時開啟小型檔案合併的選項DAG。	FALSE
hive.metastore.client.factory.class	產生實作IMetaStoreClient 介面之物件的工廠類別名稱。	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，則表示任務執行 AWS 帳戶 所在的 ID。	NULL
hive.metastore.uris	URI 中繼存放區用戶端用來連線至遠端中繼存放區的漂移。	NULL
hive.optimize.ppd	開啟述詞下推的選項。	TRUE
hive.optimize.ppd.storage	開啟儲存處理器述詞下推的選項。	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

設定	描述	預設值
hive.prewarm.numcontainers	要預先暖機的 Tez 容器數量。	10
hive.stats.autogather	讓 Hive 在INSERT OVERWRITE 命令期間自動收集基本統計資料的選項。	TRUE
hive.stats.fetch.column.stats	關閉從中繼存放區擷取資料欄統計資料的選項。當資料欄數量高時，擷取資料欄統計資料可能很昂貴。	FALSE
hive.stats.gather.num.threads	partialscan 和 noscan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 StatsProvidingRecordReader （如 ORC）。	10
hive.strict.checks.cartesian.product	開啟嚴格卡氏聯結檢查的選項。這些檢查不允許卡氏產品（交叉聯結）。	FALSE
hive.strict.checks.type.safety	開啟嚴格類型安全檢查並關閉 bigint 與 string和 比較的選項double。	TRUE
hive.support.quoted.identifiers	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 表示資料欄名稱可以包含任何字元。	COLUMN

設定	描述	預設值
hive.tez.auto.reducer.parallelism	開啟 Tez 自動減少程式平行處理功能的選項。Hive 仍然會估算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估計值。	TRUE
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 任務的核心數目。	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	Tez 在您開啟自動減少程式平行處理時指定的減少程式下限。	0.25
hive.vectorized.execution.enabled	開啟查詢執行向量化模式的選項。	TRUE
hive.vectorized.execution.reduce.enabled	開啟查詢執行之減少端的引導模式的選項。	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

設定	描述	預設值
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
mapreduce.input.fileinputformat.split.maxsize	當您的輸入格式為 時，分割運算期間分割的大小上限org.apache.hadoop.hive.ql.io.CombineHiveInputFormat 。值 0 表示沒有限制。	0
tez.am.dag.cleanup.on.completion	DAG 在完成時開啟隨機資料清除的選項。	TRUE
tez.am.emr-serverless.launch.env.[*KEY*]	在 Tez AM 程序中設定*KEY*環境變數的選項。對於 Tez AM，此值會覆寫該hive.emr-serverless.launch.env.[*KEY*]值。	
tez.am.log.level	EMR Serverless 傳遞至 Tez 應用程式主檔的根記錄層級。	INFO
tez.am.sleep.time.before.exit.millis	EMR 無伺服器應該在 AM 關閉請求後的這段時間之後推送ATS事件。	0

設定	描述	預設值
tez.am.speculation.enabled	導致推測啟動較慢任務的選項。這有助於在部分任務因機器不良或緩慢而執行速度較慢時減少任務延遲。僅支援 Amazon EMR 6.10.x 及更低版本。	FALSE
tez.am.task.max.failed.attempts	在任務失敗之前，特定任務可能失敗的嘗試次數上限。此數字不會計算手動終止的嘗試次數。	3
tez.am.vertex.cleanup.height	如果所有相依頂點都完成，Tez AM 將刪除頂點隨機切換資料的距離。當值為 0 時，此功能會關閉。Amazon 6EMR.8.0 版及更新版本支援此功能。	0
tez.client.asynchronous-stop	導致 EMR Serverless 在 Hive 驅動程式結束之前推送ATS事件的選項。	FALSE
tez.grouping.max-size	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
tez.grouping.min-size	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
tez.runtime.io.sort.mb	當 Tez 排序輸出時，軟緩衝區的大小會排序。	根據 Tez 任務記憶體計算最佳值
tez.runtime.unordered.output.buffer.size-mb	如果 Tez 未直接寫入磁碟，要使用的緩衝區大小。	根據 Tez 任務記憶體計算最佳值

設定	描述	預設值
tez.shuffle-vertex-manager.max-src-fraction	在無EMR伺服器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。在目前頂點上準備排程的任務數量，會在 min-fraction 和 之間線性擴展max-fraction 。這會預設為預設值 或 tez.shuffle-vertex-manager.min-src-fraction ，以較大者為準。	0.75
tez.shuffle-vertex-manager.min-src-fraction	在無EMR伺服器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
tez.task.emr-serverless.launch.env.[*KEY*]	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hive.emr-serverless.launch.env.[*KEY*]值。	
tez.task.log.level	EMR Serverless 傳遞給 Tez 任務的根記錄層級。	INFO
tez.yarn.ats.event.flush.timeout.millis	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 執行 Hive StartJobRun 查詢API。

```
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.ql",
            "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 儲存庫中找到有關如何執行 Hive 任務的其他範例。

EMR無伺服器 Job 恢復

EMR無伺服器版本 7.1.0 及更高版本包含工作恢復能力的支援，因此它會自動重試任何失敗的工作，無需您手動輸入。工作備援的另一個好處是，如果 AZ 遇到任何問題，EMR無伺服器會將工作執行移至不同的可用區域 (AZ)。

若要啟用作業的工作復原，請設定工作的重試原則。重試原則可確保EMR無伺服器會在任何時候失敗時自動重新啟動工作。批次和串流工作都支援重試原則，因此您可以根據使用案例自訂工作恢復能力。下表比較批次和串流作業之間的工作恢復能力的行為和差異。

	批次任務	串流工作
預設行為	不會重新執行工作。	當應用程式在執行工作時建立檢查點時，一律重試執行工作。
重試點	Batch 工作沒有檢查點，因此EMR無伺服器一律會從頭開始重新執行工作。	串流任務支援檢查點，因此您可以設定串流查詢以儲存執行時期狀態，並進度至 Amazon S3 中的檢查點位置。EMR無伺服器會繼續從檢查點執行的工作。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。
重試嘗試次數上限	允許最多 10 次重試。	串流工作具有內建的鞭打防止控制功能，因此如果工作在一小時後繼續失敗，應用程式就會停止重試工作。一小時內的預設重試次數為五次。您可以將這個重試次數設定為介於 1 或 10 之間。您無法自訂最多嘗試次數。值 1 表示不重試。

當EMR無伺服器嘗試重新執行工作時，它也會使用嘗試編號為工作索引，以便您可以在各次嘗試中追蹤工作的生命週期。

您可以使用EMR無伺服器API作業或 AWS CLI 以變更工作恢復能力或查看與工作恢復能力相關的資訊。如需詳細資訊，請參閱EMR無伺服器API指南。

根據預設，EMR無伺服器不會重新執行批次工作。若要啟用批次工作的重試，請在開始批次工作執行時設定`maxAttempts`參數。此`maxAttempts`參數僅適用於批次工作。預設值為 1，表示不要重新執行工作。接受的值為 1 到 10 (含)。

下列範例示範如何在開始工作執行時指定最多 10 次嘗試次數。

```
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"
     }
}'
```

EMR如果串流作業失敗，無伺服器會無限期地重試。若要防止因為重複無法復原的失敗而造成衝擊，請使用配置串流工作重試的防止衝擊控制。maxFailedAttemptsPerHour此參數可讓您指定EMR無伺服器停止重試前一小時內允許的失敗嘗試次數上限。預設值為 5。接受的值為 1 到 10 (含)。

```
aws emr-serverless start-job-run
 --application-id <APPPLICATION_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"
     }
}'
```

您也可以使用其他作API業執行作業取得有關作業的資訊。例如，您可以將attempt參數與作業搭配使用，以取得有關特定工GetJobRun作嘗試的詳細資訊。如果未包含attempt參數，作業會傳回有關最新嘗試的資訊。

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

```
    --attempt 1
```

此ListJobRunAttempts作業會傳回與工作執行相關之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
```

此GetDashboardForJobRun作業會建立並傳回URL可用來存取工作執行UIs的應用程式。該attempt參數可以讓你得到一URL個特定的嘗試。如果未包含attempt參數，作業會傳回有關最新嘗試的資訊。

```
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

Job 備援支援也會新增EMR無伺服器工作執行重試的新事件。EMR無伺服器會在每次重試工作時發佈此事件。您可以使用此通知來追蹤工作的重試次數。如需有關事件的詳細資訊，請參閱 Amazon EventBridge 事件。

使用重試原則記錄

每次EMR無伺服器重試工作時，嘗試都會產生自己的記錄集。為確保EMR無伺服器可以成功地將這些日誌交付到 Amazon S3 和 Amazon， CloudWatch 而不會覆寫任何日誌，EMR無伺服器會在 S3 日誌路徑和日 CloudWatch 誌串流名稱的格式中新增前綴，以包含任務的嘗試編號。

以下是格式外觀的範例。

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

此格式可確保EMR無伺服器會將每次任務嘗試的所有日誌發佈到 Amazon S3 和 CloudWatch中的指定位置。如需詳細資訊，請參閱儲存記錄檔。

> **Note**
>
> EMR無伺服器只會在所有串流工作和任何已啟用重試的批次工作中使用此前置字元格式。

Metastore 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，您可以在可存取任務的中繼存放區中保留此資料表中繼資料。

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 作為中繼存放區

您可以設定 Spark 和 Hive 任務使用 AWS Glue Data Catalog 作為其中繼存放區。當您需要持續的中繼存放區或不同應用程式、服務或 共用的中繼存放區時，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，請選擇您要使用的無EMR伺服器應用程式類型。

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。

使用 SDKs或 時 AWS CLI，您可以在任務執行的`sparkSubmit`參數`com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory`中將`spark.hadoop.hive.metastore.client.factory.class`組態設定為 。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.AWSGl
 --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 Data Catalog 中記錄中繼存放區資訊。您不需要虛擬私有雲端 （VPC） 即可使用資料目錄作為中繼存放區。

若要存取 Hive 中繼存放區資料表，請新增設定 AWS Glue IAM許可 中概述的必要 AWS Glue 政策。

設定無EMR伺服器和 AWS Glue Data Catalog 的跨帳戶存取

若要設定 EMR Serverless 的跨帳戶存取權，您必須先登入下列 AWS 帳戶：

- AccountA – 您已建立 EMR Serverless 應用程式的 AWS 帳戶 。
- 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連接至 中的資料目錄AccountB。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。

```
{
  "Version" : "2012-10-17",
  "Statement" : [ {
    "Effect" : "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::accountA:role/job-runtime-role-A"
        ]},
    "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:region:AccountB:catalog"]
  } ]
}
```

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。

```
{
  "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:region:AccountB:catalog"]
    }
  ]
}
```

3. 啟動您的任務執行。根據 AccountA的 EMR Serverless 應用程式類型，此步驟略有不同。

   Spark

   在hive-site分類中設定 spark.hadoop.hive.metastore.glue.catalogid 屬性，如下列範例所示。Replace (取代) *AccountB -catalog-id* 在 中使用 Data Catalog 的 IDAccountB。

```
aws emr-serverless start-job-run \
--application-id "application-id" \
--execution-role-arn "job-role-arn" \
--job-driver '{
    "sparkSubmit": {
        "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": {
            "spark.hadoop.hive.metastore.glue.catalogid": "AccountB-catalog-id"
```

```
        }
    }]
}'
```

Hive

在hive-site分類中設定 hive.metastore.glue.catalogid 屬性，如下列範例所示。Replace (取代) *AccountB -catalog-id* 在 中使用 Data Catalog 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 AWS Glue Data Catalog 時的考量

您可以在 Hive 指令碼JARsADD JAR中使用 新增輔助。如需其他考量，請參閱使用 AWS Glue Data Catalog 時的考量。

使用外部 Hive 中繼存放區

您可以設定EMR您的 Serverless Spark 和 Hive 任務以連線至外部 Hive 中繼存放區，例如 Amazon Aurora 或 Amazon RDS for My SQL。本節說明如何設定 Amazon RDS Hive 中繼存放區、設定您的 VPC，以及設定無EMR伺服器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 中的指示，使用私有子網路建立 VPCAmazon Virtual Private Cloud （Amazon VPC）。
2. 使用新的 Amazon VPC和私有子網路建立EMR您的 Serverless 應用程式。當您使用 設定無EMR伺服器應用程式時VPC，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 的詳細資訊VPC，請參閱 設定VPC存取權。
3. 在 Amazon 的私有子網路中建立 MySQL 或 Aurora PostgreSQL 資料庫VPC。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. 按照修改 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，以允許從無EMR伺服器安全群組JDBC進行連線。從其中一個無EMR伺服器RDS安全群組將傳入流量規則新增至安全群組。

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

設定 Spark 選項

使用 JDBC

若要設定您的 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 上的 Hive 中繼存放區互動的 Spark 入門指令碼RDS。

```
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。若要執行此操作，請在現有 Amazon EMR叢集的主節點上執行 thrift 伺服器。如果您已經擁有具有要用來簡化 Serverless 任務組態之 thrift EMR 伺服器的 Amazon EMR叢集，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/"
        }
    }
}'
```

下列程式碼範例是入門指令碼（`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:thift-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個寫入。如果您在兩個任務之間共用中繼存放區資訊，除非您寫入相同的中繼存放區資料表的不同分割區，否則請確定您不會同時寫入相同的中繼存放區資料表。

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 M連線到 Hive 中繼存放區ySQLinstance。若要執行此操作，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，以便 EMR Serverless 存取遠端 thrift 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:thirft-server-port"
    }
}
```

使用外部中繼存放區時的考量事項

- 您可以設定與 MariaDB 相容的資料庫JDBC作為您的中繼存放區。這些資料庫的範例RDS適用於 MariaDB 、我的 SQL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。您無法將 thrift 中繼存放區伺服器與 Kerberos 身分驗證搭配 EMR Serverless Spark 或 Hive 任務使用。

在另一個存取 S3 資料 AWS 來自EMR無伺服器的帳戶

您可以從一個方式執行 Amazon EMR 無伺服器任務 AWS 帳戶並將其設定為存取屬於另一個儲存貯體的 Amazon S3 儲存貯體中的資料 AWS 帳戶。本頁說明如何設定從EMR無伺服器對 S3 的跨帳戶存取。

在EMR無伺服器上執行的任務可以使用 S3 儲存貯體政策或假定角色從不同的角色存取 Amazon S3 中的資料 AWS 帳戶。

必要條件

若要為 Amazon EMR 無伺服器設定跨帳戶存取，您必須在登入兩個任務時完成任務 AWS 帳戶：

- **AccountA**— 這就是 AWS 您已建立 Amazon EMR 無伺服器應用程式的帳戶。在設定跨帳戶存取權之前，您必須在此帳戶中準備好下列項目：
  - 您想要在其中執行任務的 Amazon EMR 無伺服器應用程式。

- 具有在應用程式中執行工作所需權限的工作執行角色。如需詳細資訊，請參閱Amazon EMR Serverless 的任務執行期角色。
- **AccountB**— 這就是 AWS 包含您希望 Amazon EMR 無伺服器任務存取的 S3 儲存貯體的帳戶。

使用 S3 儲存貯體政策存取跨帳戶 S3 資料

若要存取 S3 儲存貯體 account B from account A，將下列政策附加到 S3 儲存貯體 account B.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Example permissions 1",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountA:root"
         },
         "Action": [
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::bucket_name_in_AccountB"
         ]
      },
      {
         "Sid": "Example permissions 2",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountA:root"
         },
         "Action": [
            "s3:PutObject",
            "s3:GetObject",
            "s3:DeleteObject"
         ],
         "Resource": [
            "arn:aws:s3:::bucket_name_in_AccountB/*"
         ]
      }
   ]
}
```

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 簡單儲存貯體服務使用者指南中的範例 2：儲存貯體擁有者授予跨帳戶儲存貯體許可。

使用假定角色存取跨帳戶 S3 資料

另一種為 Amazon EMR 無伺服器設定跨帳戶存取的方法是使AssumeRole用 AWS Security Token Service (AWS STS). AWS STS 是一項全域 Web 服務，可讓您為使用者請求臨時、有限權限的憑證。您可以使用您建立的臨時安全登入資料API呼叫EMR無伺服器和 Amazon S3。AssumeRole

下列步驟說明如何使用假定角色從EMR無伺服器存取跨帳戶 S3 資料：

1. 創建一個 Amazon S3 存儲桶，*cross-account-bucket*，在中AccountB。如需詳細資訊，請參閱 Amazon 簡單儲存服務使用者指南中的建立儲存貯體。如果想要擁有對 DynamoDB 的跨帳戶存取權，也可以在 AccountB 中建立 DynamoDB 資料表。如需詳細資訊，請參閱 Amazon DynamoDB 開發人員指南中的建立 DynamoDB 表格。
2. 在中建立可AccountB存取的Cross-Account-Role-BIAM角色 *cross-account-bucket*.
   a. 登入 AWS Management Console 然後在開啟IAM主控台https://console.aws.amazon.com/iam/。
   b. 選擇角色，並建立一個新角色 Cross-Account-Role-B。如需有關如何建立IAM角色的詳細資訊，請參閱《IAM使用指南》中的〈建立IAM角色〉。
   c. 建立指定存取權限Cross-Account-Role-B的IAM策略 *cross-account-bucket* S3 儲存貯體，如下列政策聲明所示。然後將IAM原則附加至Cross-Account-Role-B。如需詳細資訊，請參閱《使用指南》中的IAM〈建立IAM策略〉。

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

如果您需要 DynamoDB 存取權，請建立一個IAM政策，以指定存取跨帳戶 DynamoDB 表的權限。然後將IAM原則附加至Cross-Account-Role-B。如需詳細資訊，請參閱 Amazon DynamoDB：允許存取IAM使用者指南中的特定表格。

以下是允許存取 DynamoDB 表格的政策。CrossAccountTable

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

3. 編輯 Cross-Account-Role-B 角色的信任關係。
   a. 若要設定角色的信任關係，請為您在步驟 2 中建立的角色選擇IAM主控台中Cross-Account-Role-B的 [信任關係] 索引標籤。
   b. 選取編輯信任關係。
   c. 新增下列原則文件。這允許Job-Execution-Role-A在中AccountA擔任該Cross-Account-Role-B角色。

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

4. 授予 Job-Execution-Role-A AccountA AWS STS AssumeRole允許假設Cross-Account-Role-B。
   a. 在IAM控制台中 AWS 帳戶AccountA中，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。

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

假定的角色示例

您可以使用單一假定角色存取帳戶中的所有 S3 資源，或者使用 Amazon EMR 6.11 及更高版本，您可以設定多個IAM角色，以在存取不同的跨帳戶 S3 儲存貯體時承擔。

主題

- 使用一個假定角色存取 S3 資源
- 存取具有多個假定角色的 S3 資源

使用一個假定角色存取 S3 資源

> **Note**
>
> 當您將任務設定為使用單一假定角色時，整個任務中的所有 S3 資源都會使用該角色，包括指entryPoint令碼。

如果您想要使用單一假定角色來存取帳戶 B 中的所有 S3 資源，請指定下列組態：

1. fs.s3.customAWSCredentialsProvider將EMRFS模型組態指定為spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRole

2. 對於 Spark，請使用spark.emr-serverless.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN和spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN指定驅動程式和執行程式上的環境變數。
3. 對於 Hive，請使用hive.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARNtez.am.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN、和tez.task.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN來指定 Hive 驅動程式、Tez 應用程式主機和 Tez 工作容器上的環境變數。

下列範例顯示如何使用假定角色，以跨帳戶存取啟動EMR無伺服器工作執行。

Spark

下列範例顯示如何使用假定角色，透過跨帳戶存取 S3 來啟動EMR無伺服器 Spark 工作執行。

```
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":
 "spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRoleAWSCredentials
                "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

下列範例顯示如何使用假定角色，透過跨帳戶存取 S3 來啟動EMR無伺服器 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"
            }
        }]
    }'
```

存取具有多個假定角色的 S3 資源

在EMR無伺服器版本 6.11.0 及更新版本中，您可以設定多個IAM角色，以便在存取不同的跨帳戶值區時採用。如果您想要在帳戶 B 中使用不同的假定角色存取不同的 S3 資源，請在開始任務執行時使用下列組態：

1. `fs.s3.customAWSCredentialsProvider`將EMRFS模型組態指定為`com.amazonaws.emr.serverless.credentialsprovider.BucketLevelAssumeRoleCredent`
2. 指定EMRFS組態，`fs.s3.bucketLevelAssumeRoleMapping`以定義從 S3 儲存貯體名稱到要假設的帳戶 B 中IAM角色的對應。該值的格式應為`bucket1->role1;bucket2->role2`。

例如，您可以使用存arn:aws:iam::*AccountB*:role/Cross-Account-Role-B-1取值區bucket1，以及用arn:aws:iam::*AccountB*:role/Cross-Account-Role-B-2來存取值區bucket2。下列範例顯示如何透過多個假定角色以跨帳戶存取來啟動EMR無伺服器工作執行。

Spark

下列範例顯示如何使用多個假定角色來建立EMR無伺服器 Spark 工作執行。

```
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

下列範例顯示如何使用多個假定角色來建立EMR無伺服器 Hive 工作執行。

```
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"
            }
        }]
    }'
```

疑難排解EMR無伺服器的錯誤

使用下列資訊協助診斷和修正使用 Amazon EMR 無伺服器時可能會遇到的常見問題。

主題

- 錯誤:超過允許容量上限。
- 錯誤：已超過設定的最大容量。請稍後再試。
- 錯誤：S3 存取被拒絕。請檢查所需 S3 資源上任務執行階段角色的 S3 存取權限。
- 錯誤: ModuleNotFoundError: 沒有命名的模組<module>。有關如何在EMR無服務器中使用 python 庫，請參閱用戶指南。
- 錯誤：無法承擔執行角色，<role name>因為它不存在或未使用必要的信任關係進行設定。

錯誤:超過允許容量上限。

此錯誤表示無EMR伺服器無法提交工作，因為應用程式已超過您設定的容量上限。增加應用程式的最大容量限制。

錯誤：已超過設定的最大容量。請稍後再試。

此錯誤表示無EMR伺服器無法啟動新工作，因為應用程式已超過您設定的容量上限。增加應用程式的最大容量限制。

錯誤：S3 存取被拒絕。請檢查所需 S3 資源上任務執行階段角色的 S3 存取權限。

此錯誤表示您的任務無法存取 S3 資源。確認工作執行階段角色具有存取工作需要使用的 S3 資源的權限。若要進一步瞭解執行階段角色，請參閱Amazon EMR Serverless 的任務執行期角色。

錯誤: ModuleNotFoundError: 沒有命名的模組<module>。有關如何在EMR無服務器中使用 python 庫，請參閱用戶指南。

這個錯誤表明一個 Python 模塊不可用於星火作業。檢查相依的 Python 程式庫是否可供工作使用。如需如何封裝 Python 程式庫的詳細資訊，請參閱搭配EMR無伺服器使用 Python 程式庫。

錯誤：無法承擔執行角色，<role name>因為它不存在或未使用必要的信任關係進行設定。

此錯誤表示您為工作指定的工作執行階段角色不存在，或者該角色沒有EMR無伺服器權限的信任關係。若要驗證IAM角色是否存在，並驗證您是否已正確設定角色的信任原則，請參閱中的指示Amazon EMR Serverless 的任務執行期角色。

透EMR過 Studio 以EMR無伺服器執行互動式工作負載

概觀

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

EMR無伺服器中互動式應用程式的使用案例包括：

- 資料工程師使用 EMR Studio 中的IDE經驗來建立ETL指令碼。指令碼會從現場部署擷取資料、轉換資料以進行分析，然後將資料存放在 Amazon S3。
- 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，以偵測資料集中的異常情況。
- 資料分析師會探索資料集並建立可產生每日報告的指令碼，以更新商業儀表板等應用

必要條件

若要搭配EMR無伺服器使用互動式工作負載，您必須符合下列需求：

- EMRAmazon EMR 6.14.0 及更高版本支援無伺服器互動式應用程式。
- 若要存取互動式應用程式、執行您提交的工作負載，以及從 EMR Studio 執行互動式筆記本，您需要特定的權限和角色。如需詳細資訊，請參閱互動式工作負載所需權限。

互動式工作負載所需權限

除了存取EMR無伺服器所需的基本權限之外，您還必須為IAM身分或角色設定其他權限：

存取您的互動式應用程式

設定 EMR Studio 的使用者和工作區權限。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。

執行透過EMR無伺服器提交的工作負載

設定工作執行時期角色。如需詳細資訊，請參閱建立任務執行期角色。

若要從 EMR Studio 執行互動式筆記本

將下列其他權限新增至 Studio 使用者的IAM原則：

- **emr-serverless:AccessInteractiveEndpoints**-授予訪問和連接到您指定為的交互式應用程序的權限Resource。需要此權限才能從 EMR Studio 工作區附加至EMR無伺服器應用程式。
- **iam:PassRole**-授與存取您計劃在附加至應用程式時使用的IAM執行角色的權限。從 EMR Studio 工作區附加至EMR無伺服器應用程式需要適當的PassRole權限。

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

配置互動式應用

使用下列高階步驟建立具有來自 Amazon EMR Studio 的互動式功能的EMR無伺服器應用程式 AWS Management Console.

1. 依照中的步驟Amazon EMR Serverless 入門建立應用程式。
2. 然後，從 EMR Studio 啟動工作區，並作為運算選項連接到EMR無伺服器應用程式。如需詳細資訊，請參閱EMR無伺服器入門文件的步驟 2 中的互動式工作負載索引標籤。

當您將應用程式附加到 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。您也可以預先啟動應用程式，並在將應用程式附加至工作區之前準備就緒。

互動式應用程式考量

- EMRAmazon EMR 6.14.0 及更高版本支援無伺服器互動式應用程式。
- EMRStudio 是唯一與EMR無伺服器互動式應用程式整合的用戶端。EMR無伺服器互動式應用程式不支援下列 EMR Studio 功能：工作區共同作業、SQL總管，以及筆記本的程式設計執行。
- 只有 Spark 引擎才支援互動式應用程式。
- 交互式應用程序支持 Python 3 PySpark 和星火斯卡拉內核。
- 您可以在單一互動式應用程式上同時執行 25 筆記本。
- 沒有支援具有互動式應用程式的自我託管 Jupyter 筆記本的端點或API介面。
- 為了獲得最佳化的啟動體驗，我們建議您為驅動程式和執行程式設定預先初始化容量，並預先啟動應用程式。當您預先啟動應用程式時，請確保應用程式在您想要將其附加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、驅動程式和執行程式的預先初始化容量，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無伺服器會為每個預先初始化的驅動程式維護預先初始化 根據預設，即使您未為驅動程式指定任何預先初始化容量，EMRServerless 仍會在整個應用程式中維護一個核心工作站的預先初始化容量。每個核心工作者都使用 4 v CPU 和 16 GB 的記憶體。如需目前的定價資訊，請參閱 Amazon EMR 定價頁面。
- 您必須擁有足夠的 v CPU 服務配額 AWS 帳戶 以執行互動式工作負載。如果您未執行啟用 Lake 格式化的工作負載，建議至少 24 v。CPU如果這樣做，我們建議至少 28 v CPU。
- EMR如果筆記型電腦閒置超過 60 分鐘，無伺服器會自動終止核心。EMR無伺服器會計算筆記型電腦工作階段期間上次完成活動的核心閒置時間。您目前無法修改核心閒置逾時設定。
- 若要啟用具有互動式工作負載的 Lake Formation，請在建立EMR無伺服器應用程式時，`spark.emr-serverless.lakeformation.enabled`將組態設定為在`runtime-configuration`物件中的`spark-defaults`分類`true`下。要了解有關在EMR無伺服器中啟用 Lake Formation 的更多信息，請參閱在 Amazon EMR 中啟用 Lake Formation。

透過 Apache Livy 端點使用EMR無伺服器執行互動式工作負載

使用 Amazon 6.14.0 及更高EMR版本，您可以建立並啟用 Apache Livy 端點，同時建立EMR無伺服器應用程式，並透過自我託管筆記本或自訂用戶端執行互動式工作負載。Apache 利維端點提供以下優點：

- 您可以透過 Jupyter 筆記本安全地連線到 Apache Livy 端點，並透過 Apache Livy 的介面管理 Apache Spark 工作負載。REST
- 針對使用來自 Apache Spark 工RESTAPI作負載資料的互動式 Web 應用程式，使用 Apache Livy 作業。

必要條件

若要搭配EMR無伺服器使用 Apache Livy 端點，您必須符合下列需求：

- 完成開始使用 Amazon EMR 無伺服器中的步驟。
- 若要透過 Apache Livy 端點執行互動式工作負載，您需要特定的權限和角色。如需詳細資訊，請參閱互動式工作負載的必要權限。

所需的許可

除了存取EMR無伺服器的必要權限之外，您還必須將下列權限新增至您的IAM角色，才能存取 Apache Livy 端點並執行應用程式：

- `emr-serverless:AccessLivyEndpoints`— 授予訪問權限，並連接到您指定為啟用 LIVE 的應用程序。`Resource`您需要此權限才能執行可從 Apache Livy 端點執行可用的RESTAPI作業。
- `iam:PassRole`— 授予在建立 Apache Livy 工作階段時存取IAM執行角色的權限。EMR無伺服器將使用此角色來執行您的工作負載。
- `emr-serverless:GetDashboardForJobRun`— 授予產生 Spark Live UI 和驅動程式記錄連結的權限，並提供存取記錄，做為 Apache Livy 工作階段結果的一部分。

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

```
            "Action": "emr-serverless:AccessLivyEndpoints",
            "Resource": "arn:aws:emr-serverless:<AWS_REGION>:account:/applications/*"
        },
        {
            "Sid": "EMRServerlessRuntimeRoleAccess",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "execution-role-ARN",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "emr-serverless.amazonaws.com"
                }
            }
        },
        {
            "Sid": "EMRServerlessDashboardAccess",
            "Effect": "Allow",
            "Action": "emr-serverless:GetDashboardForJobRun",
            "Resource": "arn:aws:emr-serverless:<AWS_REGION>:account:/applications/*"
        }
    ]
}
```

開始使用

1. 要創建一個 Apache 的 Livy 啟用的應用程序，運行以下命令。

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

2. EMR無伺服器建立應用程式後，啟動應用程式以使 Apache Livy 端點可用。

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

使用下面的命令來檢查你的應用程序的狀態。一旦狀態變成STARTED，您就可以存取 Apache Livy 端點。

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

```
--region <AWS_REGION> --application-id >application_id>
```

3. 使用以下URL指令存取端點：

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

端點準備就緒後，您可以根據您的使用案例提交工作負載。您必須使用SIGv4協議將每個請求簽署到端點，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 — 您必須使用自訂HTTP用戶端提交 Apache Livy 端點API作業。
- Sparkmagic 核心 — 您必須在本機執行 Sparkmagic 核心，並使用 Jupyter 筆記本提交互式查詢。

HTTP客戶端

若要建立 Apache Livy 工作階段，您必須`emr-serverless.session.executionRoleArn`在要求主體的`conf`參數中提交。下列範例是要`POST /sessions`求範例。

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

下表描述了所有可用的 Apache 利維API操作。

API操作	描述
GET/會話	傳回所有作用中互動式工作階段的清單。
POST/會話	通過火花或 pyspark 創建一個新的交互式會話。
GET/會議/ <*sessionId* >	返回會話信息。
GET/會議/ <*sessionId* >/ 州	返回會話的狀態。
DELETE/會議/ <*sessionId* >	停止並刪除工作階段。

API操作	描述
GET/會議/ <*sessionId* > /聲明	返回會話中的所有語句。
POST/會議/ <*sessionId* > /聲明	在工作階段中執行陳述式。
GET/會議/ <*sessionId* >/ 月結單 / <*statementId* >	返回會話中指定語句的詳細信息。
POST/會議/ <*sessionId* >/ 月結單 / <*statementId* >/ 取消	取消此會話中指定的語句。

將請求發送到阿帕奇利維端點

您也可以從HTTP用戶端將要求直接傳送到 Apache Livy 端點。這樣做可讓您在筆記型電腦以外的地方遠端執行使用案例的程式碼。

在開始向端點傳送要求之前，請確定您已安裝下列程式庫：

```
pip3 install botocore awscrt requests
```

以下是將HTTP請求直接發送到端點的示例 Python 腳本：

```
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())
```

火花核心

在安裝火花之前，請確保您已配置 AWS 您要在其中安裝 sparkmagic 的執行個體中的認證

1. 按照安裝步驟安裝火花。請注意，您只需要執行前四個步驟。
2. sparkmagic 核心支援自訂驗證器，因此您可以將驗證器與 sparkmagic 核心整合，以便簽署每個要求。SIGv4
3. 安裝EMR無伺服器自訂驗證器。

   ```
   pip install emr-serverless-customauth
   ```
4. 現在，在 Sparkmagic 設定 json 檔案URL中提供自訂驗證器和 Apache Livy 端點的路徑。使用以下命令打開配置文件。

   ```
   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. 啟動木普特實驗室。它應該使用您在最後一步中設置的自定義身份驗證。
6. 然後，您可以運行以下筆記本命令和代碼以開始使用。

```
%%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，每個指令都會透過設定的 Apache Livy 端點呼叫每個 Apache Livy API 作業。URL然後，您可以根據您的用例編寫說明。

考量事項

透過 Apache Livy 端點執行互動式工作負載時，請考慮下列考量事項。

- EMR無伺服器會使用呼叫者主體來維護工作階段層級隔離。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要項目。如需更精細的隔離，您可以在採用認證時設定來源身分識別。在此情況下，EMRServerless 會根據呼叫者主體和來源識別強制執行工作階段層級隔離。如需來源身分識別的詳細資訊，請參閱監視和控制對假定角色採取的動作。
- EMR無伺服器版本 6.14.0 及更新版本支援 Apache Livy 端點。
- 阿帕奇利維端點僅支持 Apache 星火引擎。
- 阿帕奇生活端點支持斯卡拉星火和 PySpark。
- 依預設，會`autoStopConfig`在您的應用程式中啟用。這意味著應用程序在閒置 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 端點的單一應用程式上執行多達 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程式設定預先初始化的容量。
- 您必須先手動啟動應用程式，才能連線到 Apache Livy 端點。
- 您必須擁有足夠的 v CPU 服務配額 AWS 帳戶 以使用 Apache Livy 端點執行互動式工作負載。我們建議至少 24 v CPU。
- 預設的 Apache 利維工作階段逾時為 1 小時。如果您沒有一小時執行陳述式，則 Apache Livy 會刪除工作階段並釋放驅動程式和執行程式。您無法變更此設定。
- 只有作用中的工作階段可以與 Apache Livy 端點互動。工作階段完成、取消或終止後，您將無法透過 Apache Livy 端點存取它。

日誌記錄和監控

監控是維護無EMR伺服器應用程式和任務的可靠性、可用性和效能的重要部分。您應該從無EMR伺服器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

若要在 EMR Serverless 上監控您的任務進度並疑難排解任務失敗，您可以選擇 EMR Serverless 儲存和提供應用程式日誌的方式。當您提交任務執行時，您可以將受管儲存、Amazon S3 和 Amazon 指定 CloudWatch 為記錄選項。

透過 CloudWatch，您可以指定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 記錄時，下表顯示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	應用程式使用者介面
受管儲存	存放在受管儲存體中	存放在受管儲存體中	支援
受管儲存和 S3 儲存貯體	存放在兩個位置	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 for EMR Serverless

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

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
        }
    }
}
```

使用 Amazon S3 儲存貯體記錄無EMR伺服器

在任務將日誌資料傳送至 Amazon S3 之前，您必須在任務執行期角色的許可政策中包含下列許可。*amzn-s3-demo-logging-bucket* 以您的記錄儲存貯體名稱取代 。

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

```
    ]
}
```

若要設定 Amazon S3 儲存貯體以儲存來自 的日誌 AWS CLI，請在開始任務執行時使用 `s3MonitoringConfiguration` 組態。若要這麼做，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，EMRServerless 會將日誌傳送至下列路徑：

```
'/applications/<applicationId>/jobs/<jobId>'
```

EMR 無伺服器 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。如果您在啟用重試時執行任務，EMRServerless 會自動將嘗試次數新增至日誌路徑字首，以便您更妥善地區分和追蹤日誌。

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

使用 Amazon 記錄無EMR伺服器 CloudWatch

當您將任務提交至 EMR Serverless 應用程式時，您可以選擇 Amazon CloudWatch 作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 日誌分析功能。您也可以將日誌從 串流 CloudWatch 到其他系統，例如 OpenSearch 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 即時尾部功能或透過 CloudWatch CLI尾部命令即時檢視日誌。

根據預設， CloudWatch 記錄會針對無EMR伺服器停用。若要啟用它，請參閱 中的組態AWS CLI。

> **Note**
>
> Amazon 會即時 CloudWatch 發佈日誌，因此從工作者身上產生更多資源。如果您選擇低工作者容量，可能會對任務執行時間造成的影響增加。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 的每秒交易 （TPS） 速率太低，日誌發佈也可能會限

流PutLogEvents。 CloudWatch 限流組態對所有 服務都是全域的，包括無EMR伺服器。如需詳細資訊，請參閱如何在 re：post 上判斷 CloudWatch 日誌中的限流？。 AWS

使用 記錄的必要許可 CloudWatch

在任務將日誌資料傳送至 Amazon 之前 CloudWatch，您必須在任務執行期角色的許可政策中包含下列許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:AWS ##:111122223333:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
               "logs:PutLogEvents",
               "logs:CreateLogGroup",
               "logs:CreateLogStream",
               "logs:DescribeLogStreams"
            ],
            "Resource": [
               "arn:aws:logs:AWS ##:111122223333:log-group:my-log-group-name:*"
            ]
        }
    ]
}
```

AWS CLI

若要設定 Amazon 從 CloudWatch 儲存無EMR伺服器日誌 AWS CLI，請在開始任務執行時使用 `cloudWatchLoggingConfiguration` 組態。若要這麼做，請提供下列組態覆寫。您也可以選擇性地提供日誌群組名稱、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 會使用預設日誌串流 /aws/emr-serverless將日誌 CloudWatch 發佈至預設日誌群組 /applications/*applicationId*/jobs/*jobId*/*worker-type*。

EMR 無伺服器 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。如果您為任務啟用重試，EMRServerless 會自動將嘗試次數新增至日誌路徑字首，以便您更妥善地區分和追蹤日誌。

```
'/applications/<applicationId>/jobs/<jobId>/attempts/<attemptNumber>/worker-type'
```

下列顯示使用 EMR Serverless 預設設定開啟 Amazon CloudWatch 日誌記錄所需的最低組態：

```
{
    "monitoringConfiguration": {
        "cloudWatchLoggingConfiguration": {
            "enabled": true
         }
     }
}
```

下列範例顯示您在開啟 Amazon CloudWatch logging for EMR Serverless 時可以指定的所有必要和選用組態。支援的 logTypes 值也會列在此範例下方。

```
{
    "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，EMRServerless 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，則可以使用 logTypes 欄位指定容器角色和對應的日誌類型。

下列清單顯示您可以為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並減少任務失敗的數量，因為您磁碟上沒有更多空間。

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版本標籤。

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版本標籤。

EMR Serverless 也會輪換驅動程式和執行程式的 Spark 應用程式日誌，例如 `stdout`和 `stderr` 檔案。您可以使用 Spark History Server 和 Live UI 連結，在 Studio 中選擇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。若要查看較舊的輪換日誌，您必須在儲存日誌時指定 Amazon S3 位置。如需詳細資訊，請參閱使用 Amazon S3 儲存貯體記錄無EMR伺服器。

您可以在下列位置找到最新的日誌檔案。EMR 無伺服器每 15 秒重新整理檔案。這些檔案的範圍從 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 執行程式。此變更僅適用於 S3 記錄。日誌輪換不會對上傳至 Amazon 的日誌串流引入任何變更 CloudWatch。

EMR 無伺服器 7.1.0 版和更新版本支援串流和批次工作的重試嘗試。如果您已啟用對任務的重試嘗試，EMR則 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 加密 EMR Serverless 日誌 CloudWatch

若要 CloudWatch 使用您自己的KMS金鑰加密 Amazon 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 的加密金鑰許可 CloudWatch

必要的使用者許可

提交任務或檢視日誌或應用程式的使用者UIs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IAM的政策中指定許可。如果提交任務的使用者缺少KMS金鑰許可，EMRServerless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 的許可`kms:Decrypt`：

```
{
    "Effect": "Allow",
    "Principal":{
       "AWS": "arn:aws:iam::111122223333:user/user-name"
     },
     "Action": [
       "kms:GenerateDataKey",
       "kms:Decrypt"
      ],
     "Resource": "*"
 }
```

範例IAM政策

下列IAM政策提供 kms:GenerateDataKey和 的許可kms:Decrypt：

```
{
 "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
     "Action": [
           "kms:GenerateDataKey",
           "kms:Decrypt"
      ],
     "Resource": "key-arn"
   }
}
```

若要啟動 Spark 或 Tez UI，您必須授予使用者、群組或角色存取 的許可emr-serverless:GetDashboardForJobRunAPI，如下所示：

```
{
 "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
     "Action": [
           "emr-serverless:GetDashboardForJobRun"
      ]
   }
}
```

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，您必須設定KMS金鑰許可，如下所示。

`emr-serverless.amazonaws.com` 主體必須在 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，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 使用KMS金鑰ARN。

任務執行期角色在其IAM政策中必須具有下列許可：

```
{
 "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
     "Action": [
           "kms:GenerateDataKey",
           "kms:Decrypt"
      ],
     "Resource": "key-arn"
   }
}
```

Amazon 的加密金鑰許可 CloudWatch

若要將KMS金鑰ARN與日誌群組建立關聯，請針對任務執行期角色使用下列IAM政策。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "logs:AssociateKmsKey"
        ],
        "Resource": [
            "arn:aws:logs:AWS ##:111122223333:log-group:my-log-group-name:*"
        ]
    }
}
```

設定KMS金鑰政策以授予 Amazon KMS許可 CloudWatch：

```
{
    "Version": "2012-10-17",
    "Id": "key-default-1",
    "Statement":
     {
        "Effect": "Allow",
        "Principal": {
            "Service": "logs.AWS ##.amazonaws.com"
        },
        "Action": [
            "kms:Decrypt",
            "kms:GenerateDataKey",
        ],
        "Resource": "*",
        "Condition": {
            "ArnLike": {
                "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:AWS #
#:111122223333:*"
            }
        }
     }
}
```

設定 Amazon EMR Serverless 的 Apache Log4j2 屬性

此頁面說明如何在 設定無EMR伺服器任務的自訂 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

使用 Amazon 6.8.0 版和更新EMR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對 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

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`

此分類會設定驅動程式`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"
                }
            }
       ]
    }'
```

Spark 任務範例中的 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，您可以使用下列組態：

```
{
   "classification": "spark-driver-log4j2",
      "properties": {
          "rootLogger.level":"error", // only display Spark error logs
          "logger.PySparkApp.level": "info",
          "logger.PySparkApp.name": "PySparkApp"
      }
}
```

Scala

Example - 將 Log4j2 用於 Spark 任務與 Scala

```
import org.apache.log4j.Logger
import org.apache.spark.sql.SparkSession

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，您可以使用下列組態：

```
{
   "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

下列 Log4j2.x 屬性無法針對 Spark 程序設定：

- `rootLogger.appenderRef.stdout.ref`
- `appender.console.type`
- `appender.console.name`
- `appender.console.target`
- `appender.console.layout.type`
- `appender.console.layout.pattern`

如需有關您可以設定之 Log4j2.x 屬性的詳細資訊，請參閱 上的 [log4j2.properties.template 檔案](#) GitHub。

監控無EMR伺服器

本節涵蓋您可以監控 Amazon EMR Serverless 應用程式和任務的方式。

主題

監控無EMR伺服器應用程式和任務

透過 Amazon 無EMR伺服器 CloudWatch 指標，您可以接收 1 分鐘 CloudWatch 的指標和存取 CloudWatch 儀表板，以檢視 near-real-time無EMR伺服器應用程式的操作和效能。

EMR Serverless CloudWatch 每分鐘都會傳送指標給 。EMR 無伺服器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 EMR GitHub 儲存庫中提供的 Serverless CloudWatch 儀表板範本並部署。

> **Note**
>
> EMR 無伺服器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，您可以從 EMR Studio Workspace 中檢視日誌和 Apache Spark UI。

下表說明`AWS/EMRServerless`命名空間中可用的無EMR伺服器維度。

EMR Serverless 指標的維度

維度	描述	
`ApplicationId`	篩選 EMR Serverless 應用程式的所有指標。	
`JobId`	篩選無EMR伺服器任務執行的所有指標。	

維度	描述	
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_EXECUTORS SPARK_DRIVER 和 。	
CapacityAllocationType	篩選指定容量配置類型的所有指標。例如，您可以篩選 PreInitCapacity 的預先初始化容量和其他OnDemandCapacity 所有項目。	

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用量。您也可以設定單一檢視來監控 CloudWatch 儀表板中的應用程式容量用量。

EMR 無伺服器應用程式指標

指標	描述	主要維度	次要維度
CPUAllocated	vCPUs 已配置的總數。	ApplicationId	ApplicationId ，WorkerType ，CapacityAllocationType
IdleWorkerCount	工作者閒置總數。	ApplicationId	ApplicationId ，WorkerType ，CapacityAllocationType
MaxCPUAllowed	應用程式CPU允許的最大值。	ApplicationId	N/A
MaxMemoryAllowed	應用程式允許的 GB 記憶體上限。	ApplicationId	N/A

指標	描述	主要維度	次要維度
MaxStorageAllowed	應用程式允許的 GB 儲存容量上限。	ApplicationId	N/A
MemoryAllocated	GB 配置的總記憶體。	ApplicationId	ApplicationId ,WorkerType ,CapacityAllocationType
PendingCreationWorkerCount	待建立的工作者總數。	ApplicationId	ApplicationId ,WorkerType ,CapacityAllocationType
RunningWorkerCount	應用程式正在使用的工作者總數。	ApplicationId	ApplicationId ,WorkerType ,CapacityAllocationType
StorageAllocated	以 GB 為單位配置的總磁碟儲存體。	ApplicationId	ApplicationId ,WorkerType ,CapacityAllocationType
TotalWorkerCount	可用的工作者總數。	ApplicationId	ApplicationId ,WorkerType ,CapacityAllocationType

任務層級監控

Amazon EMR Serverless Amazon CloudWatch 每一分鐘會傳送下列任務層級指標。您可以依任務執行狀態檢視彙總任務執行的指標值。每個指標的單位都是計數 。

EMR 無伺服器任務層級指標

指標	描述	主要維度
SubmittedJobs	處於已提交狀態的任務數量。	ApplicationId
PendingJobs	處於擱置狀態的任務數目。	ApplicationId
ScheduledJobs	排程狀態中的任務數目。	ApplicationId
RunningJobs	處於執行中狀態的任務數量。	ApplicationId
SuccessJobs	處於成功狀態的任務數目。	ApplicationId
FailedJobs	處於失敗狀態的任務數目。	ApplicationId
CancellingJobs	處於取消狀態的任務數量。	ApplicationId
CancelledJobs	處於已取消狀態的任務數目。	ApplicationId

您可以使用引擎特定應用程式 來監控執行中和已完成的無EMR伺服器任務的引擎特定指標UIs。當您檢視執行中任務的 UI 時，您會看到即時應用程式 UI 與即時更新。當您檢視已完成任務的 UI 時，您會看到持久性應用程式 UI。

執行任務

對於執行中的無EMR伺服器任務，您可以檢視提供引擎特定指標的即時介面。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 主控台或請求搭配 的安全URL端點 AWS Command Line Interface。

已完成的任務

對於已完成的無EMR伺服器任務，您可以使用 Spark 歷史記錄伺服器或持久性 Hive Tez 使用者介面來檢視 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，或透過 請求安全URL端點 AWS Command Line Interface。

任務工作者層級監控

Amazon EMR Serverless 會將命名AWS/EMRServerless空間和Job Worker Metrics指標群組中可用的下列作業工作者層級指標傳送至 Amazon CloudWatch。EMR Serverless 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作為維度，監控屬於相同應用程式的多個任務。

EMR 無伺服器任務工作者層級指標

指標	描述	單位	主要維度	次要維度
WorkerCpuAllocated	任務執行中為工作者配置的 vCPU 核心總數。	無	JobId	ApplicationId 、WorkerType 與 CapacityAllocationType
WorkerCpuUsed	作業執行中工作者使用的 vCPU 核心總數。	無	JobId	ApplicationId 、WorkerType 與 CapacityAllocationType
WorkerMemoryAllocated	任務執行中為工作者配置的總記憶體，單位為 GB。	Gigabytes（GB）	JobId	ApplicationId 、WorkerType 與 CapacityAllocationType
WorkerMemoryUsed	作業執行中工作者使用的總記憶體，單位為 GB。	Gigabytes（GB）	JobId	ApplicationId 、WorkerType 與 CapacityAllocationType
WorkerEphemeralSto	任務執行中為工作者配置的暫時性儲存的位元組數。	Gigabytes（GB）	JobId	ApplicationId 、WorkerType 與

指標	描述	單位	主要維度	次要維度
rageAlloc ated				CapacityA llocation Type
WorkerEph emeralSto rageUsed	作業執行中工作者使用的暫時性儲存的位元組數。	Gigabytes （GB）	JobId	Applicati onId 、WorkerTyp e 與 CapacityA llocation Type
WorkerSto rageReadB ytes	作業執行中工作者從儲存體讀取的位元組數。	位元組	JobId	Applicati onId 、WorkerTyp e 與 CapacityA llocation Type
WorkerSto rageWrite Bytes	任務執行中從工作者寫入儲存體的位元組數。	位元組	JobId	Applicati onId 、WorkerTyp e 與 CapacityA llocation Type

下列步驟說明如何檢視各種類型的指標。

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說明，導覽至 EMR Studio 上的無EMR伺服器應用程式。
2. 若要檢視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. 若要檢視 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。若要檢視 Hive 引擎日誌，請在 Hive Tez UI DAG 中選擇適當 的日誌連結。

3. 若要檢視已完成任務的引擎特定應用程式UIs和日誌：

a. 選擇具有 SUCCESS 狀態的任務。

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 History Server 或持久性 Hive Tez UI，以導覽至任務類型的應用程式 UI。

d. 若要檢視 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。若要檢視 Hive 引擎日誌，請在 Hive Tez UI DAG中選擇適用於 的日誌連結。

AWS CLI

使用 存取您的應用程式使用者介面 AWS CLI

- 若要產生URL可用於存取執行中和已完成任務的應用程式使用者介面的 ，請呼叫 `GetDashboardForJobRun` API。

```
aws emr-serverless get-dashboard-for-job-run /
--application-id <application-id> /
--job-run-id <job-id>
```

URL 您產生的 有效期為一小時。

使用 Amazon Managed Service for Prometheus 監控 Spark 指標

使用 Amazon 7.1.0 版和更新EMR版本，您可以將 EMR Serverless 與 Amazon Managed Service for Prometheus 整合，以收集無EMR伺服器任務和應用程式的 Apache Spark 指標。當您提交任務或使用 AWS 主控台、無EMR伺服器 或 建立應用程式時API，即可使用此整合 AWS CLI。

必要條件

您必須完成下列先決條件，才能將 Spark 指標交付至 Amazon Managed Service for Prometheus。

- 建立 Amazon Managed Service for Prometheus 工作區。此工作區用作擷取端點。記下為端點 - 遠端寫入 URL顯示的 。 URL建立 EMR Serverless 應用程式URL時，您需要指定 。
- 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 ，然後在您要設定的欄位內指定資訊來設定應用程式。
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 <APPPLICATION_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"
        }
    }
}'
```

包含在您的命令`prometheusMonitoringConfiguration`中，表示 EMR Serverless 必須使用收集 Spark 指標的代理程式來執行 Spark 任務，並將其寫入您的 Amazon Managed Service for Prometheus `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、警示和分析。

進階組態屬性

EMR Serverless 在 Spark 中使用名為 的元件`PrometheusServlet`來收集 Spark 指標，並將效能資料轉換為與 Amazon Managed Service for Prometheus 相容的資料。根據預設，當您使用 提交任務時，無EMR伺服器會在 Spark 中設定預設值，並剖析驅動程式和執行器指標`PrometheusMonitoringConfiguration`。

下表說明在提交 Spark 任務時，您可以將指標傳送至 Amazon Managed Service for Prometheus 時設定的所有屬性。

Spark 屬性	預設值	描述
spark.metrics.conf.*.sink.prometheusServlet.class	org.apache.spark.metrics.sink。PrometheusServlet	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。
spark.metrics.conf.*.source.jvm.class	org.apache.spark.metrics.source。JvmSource	類別 Spark 使用 來收集和傳送基礎 Java 虛擬機器的重要指標。若要停止收集JVM指標，請將此屬性設定為空字串來停用，例如 ""。若要覆寫預設行為，請指定您自己的自訂類別。
spark.metrics.conf.driver.sink.prometheusServlet.path	/metrics/prometheus	Amazon Managed Service for Prometheus 用於從驅動程式收集指標URL的不同之處。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 ""。
spark.metrics.conf.executor.sink.prometheusServlet.path	/metrics/executor/prometheus	URL Amazon Managed Service for Prometheus 用來從執行器收集指標的不同之處。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 。

考量與限制

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 一般可用的 。
- 執行代理程式在 Amazon Managed Service for Prometheus 上收集 Spark 指標需要工作者的更多資源。如果您選擇較小的工作者規模，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 7.1.0 版及更新EMR版本。

EMR 無伺服器用量指標

您可以使用 Amazon CloudWatch 用量指標來提供您的帳戶使用資源的可見性。使用這些指標，在 CloudWatch 圖形和儀表板上視覺化您的服務用量。

EMR 無伺服器用量指標對應至 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使用者指南中的服務配額和 Amazon CloudWatch 警示。Service Quotas

如需有關無EMR伺服器服務配額的詳細資訊，請參閱 的端點和配額 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	描述
`ResourceCount`	帳戶上執行的指定資源總數。資源由與指標相關聯的維度定義。

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 發佈的使用指標。

維度	Value	描述
Service	EMR 無伺服器	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。
Class	無	EMR Serverless 正在追蹤的資源類別。

使用 將 EMR Serverless 自動化 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。透過 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 EventBridge 搭配 EMR Serverless 使用 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 無伺服器發出三種類型的事件：

- 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 – 發出任務執行的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 版及更高版本發出任務執行的每次重試的事件。

範例無EMR EventBridge伺服器事件

EMR Serverless 報告的事件具有aws.emr-serverless指派給 的值source，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"
    }
}
```

標記 資源

您可以使用標籤為每個資源指派自己的中繼資料，以協助您管理EMR無伺服器資源。本節提供標籤功能的概觀，並說明如何建立標籤。

主題

什麼是標籤？

標籤是您指派給標籤的標籤 AWS 資源。每個標記皆包含由您定義的索引鍵和值。標籤使您可以對您的進行分類 AWS 依屬性 (例如目的、擁有者和環境) 的資源。當您有許多相同類型的資源時，您可以依據先前指派的標籤，快速識別特定的資源。例如，您可以為 Amazon EMR 無伺服器應用程式定義一組標籤，以協助您追蹤每個應用程式的擁有者和堆疊層級。建議您為每個資源類型設計一組一致的標籤金鑰。

標籤不會自動指派給您的資源。將標籤新增至資源後，您可以隨時修改標籤的值或從資源中移除標籤。標籤對 Amazon EMR 無伺服器沒有任何語義意義，並嚴格解釋為字元字串。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，您可以控制哪些使用者 AWS 帳戶具有管理標籤的權限。如需標籤型存取控制政策範例，請參閱 標籤型存取控制的政策。

標記您的 資源

您可以標記新的或現有的應用程式和工作執行。如果您使用的是 Amazon EMR 無服務器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。您可以使用`TagResource`API動作將標籤套用至現有資源。

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 無伺服器資源。

資源	支援標籤	支援標籤傳播	支援建立時標記 (Amazon EMR 無伺服器API、AWS CLI和 AWS SDK)	API用於創建（標籤可以在創建過程中添加）
應用程式	是	不。 與應用程式相關聯的標記不會傳播到提交至該應用程式的工作執行。	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：

- 每個資源最多可以有 50 個使用者建立的標籤。
- 對於每一個資源，每個標籤金鑰必須是唯一的，且每個標籤金鑰只能有一個值。
- 最大密鑰長度是 128 碼字符 UTF -8。
- 最大值長度為 UTF -8 中 256 個萬國碼字元。
- 允許的字符是字母，數字，UTF-8 中可表示的空格，以及以下字符：_。：/= +-@。
- 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，例如鍵或值的前綴。這些僅保留用於 AWS 使用。

使用標籤 AWS CLI 和 Amazon EMR 無服務器 API

使用以下內容 AWS CLI 用於新增、更新、列出和刪除資源標籤的命令或 Amazon EMR 無伺服器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`

下列範例顯示如何使用標記或取消標記資源 AWS CLI.

標記現有的應用程式

以下命令標記現有的應用程序。

```
aws emr-serverless tag-resource --resource-arn resource_ARN --tags team=devs
```

取消標記現有的應用程式

以下命令刪除現有應用程序中的標籤。

```
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無伺服器教學課程

本節說明使用EMR無伺服器應用程式時的常見使用案例。

主題

將 Java 17 與 Amazon EMR 無伺服器搭配使用

在 Amazon 6.11.0 及更高EMR版本中，您可以將EMR無伺服器 Spark 任務設定為使用 Java 17 執行階段的 Java 虛擬機器 ()。JVM使用以下方法之一來配置與 Java 17 星火。

`JAVA_HOME`

若要覆寫EMR無伺服器 6.11.0 及更新版本的JVM設定，您可以將`JAVA_HOME`設定提供給其`spark.emr-serverless.driverEnv`與`spark.executorEnv`環境分類。

x86_64

設置所需的屬性以指定 Java 17 作為 Spark 驅動程序和執行程序的`JAVA_HOME`配置：

```
--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

設置所需的屬性以指定 Java 17 作為 Spark 驅動程序和執行程序的JAVA_HOME配置：

```
--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無伺服器 6.11.0 及更新版本的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EMR配無伺服器使用 Apache 胡迪

若要搭配EMR無伺服器應用程式使用 Apache Hudi

1. 在相應的 Spark 作業執行中設定所需的 Spark 屬性。

```
spark.jars=/usr/lib/hudi/hudi-spark-bundle.jar
spark.serializer=org.apache.spark.serializer.KryoSerializer
```

2. 若要將 Hudi 表格同步至已設定的目錄，請指定 AWS Glue 資料目錄做為您的中繼存放區，或設定外部中繼存放區。EMR無伺服器支援hms做為 Hudi 工作負載 Hive 資料表的同步模式。EMR無伺服器會將此內容啟用為預設值。若要進一步瞭解如何設定中繼存放區，請參閱Metastore 組態。

   > **Important**
   >
   > EMR無伺服器不支援 HIVEQL Hive 表格的同步模式選項，也不支援處理 Hudi 工JDBC作負載的同步模式選項。若要深入瞭解，請參閱同步模式。

   當您使用 AWS Glue 資料目錄做為中繼存放區，您可以為 Hudi 工作指定下列組態屬性。

```
--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 的 Apache Hudi 發行版本EMR，請參閱 Hudi 發行歷史記錄。

將 Apache 冰山與EMR無伺服器搭配使用

將 Apache 冰山與EMR無伺服器應用程式搭配使用

1. 在相應的 Spark 作業執行中設定所需的 Spark 屬性。

```
spark.jars=/usr/share/aws/iceberg/lib/iceberg-spark3-runtime.jar
```

2. 指定 AWS Glue 資料目錄做為您的中繼存放區或設定外部中繼存放區。若要進一步瞭解如何設定中繼存放區，請參閱Metastore 組態。

   配置要用於冰山的中繼存儲屬性。例如，如果您想要使用 AWS Glue 資料目錄，在應用程式組態中設定下列屬性。

```
spark.sql.catalog.dev.warehouse=s3://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/
spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
spark.sql.catalog.dev=org.apache.iceberg.spark.SparkCatalog
spark.sql.catalog.dev.catalog-impl=org.apache.iceberg.aws.glue.GlueCatalog
spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSGl
```

   當您使用 AWS Glue 資料目錄做為您的中繼存放區，您可以為 Iceberg 工作指定下列組態屬性。

```
--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://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/
--conf
 spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSG
```

要了解有關 Apache 冰山版本的更多信息 Amazon EMR 請參閱冰山發布歷史記錄。

搭配EMR無伺服器使用 Python 程式庫

在 Amazon EMR 無伺服器應用程式上 PySpark 執行任務時，可以將各種 Python 程式庫封裝為相依性。若要這麼做，您可以使用原生 Python 功能、建置虛擬環境，或直接設定 PySpark 工作以使用 Python 程式庫。此頁面涵蓋了每種方法。

使用原生 Python 功能

當您設置以下配置時，您可 PySpark 以使用上傳 Python 文件（ `.py` ），Python 壓縮包（ `.zip` ）和雞蛋文件（ `.egg` ）到星火執行器。

```
--conf spark.submit.pyFiles=s3://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/<.py|.egg|.zip file>
```

如需如何針對 PySpark 工作使用 Python 虛擬環境的詳細資訊，請參閱使用 PySpark 原生功能。

建立虛 Python 環境

若要為一項 PySpark 工作封裝多個 Python 程式庫，您可以建立獨立的 Python 虛擬環境。

1. 要構建 Python 虛擬環境，請使用以下命令。所示範例會將套`scipy`件安裝`matplotlib`到虛擬環境套件中，然後將存檔複製到 Amazon S3 位置。

   > **Important**
   >
   > 您必須在類似的 Amazon Linux 2 環境中運行以下命令，使用與在EMR無服務器中使用相同的 Python 版本，即 Amazon EMR 6.6.0 版本的 Python 3.7.10。您可以在EMR無伺服器範例儲存庫中找到 Docker 檔案範例。 GitHub

   ```
   # 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://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/
   ```

```
# optionally, remove the virtual environment directory
rm -fr pyspark_venvsource
```

2. 提交 Spark 工作，並將您的屬性設置為使用 Python 虛擬環境。

```
--conf spark.archives=s3://DOC-EXAMPLE-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，如果您不覆蓋原始的 Python 二進製文件，則前面設置序列中的第二個配置將是`--conf spark.executorEnv.PYSPARK_PYTHON=python`。

有關如何使用 Python 虛擬環境進行 PySpark 作業的更多信息，請參閱使用虛擬環境。如需如何提交 Spark 工作的更多範例，請參閱Spark 任務。

設定 PySpark 工作以使用 Python 程式庫

使用 Amazon 6.12.0 及更高EMR版本，您可以直接設定EMR無伺服器任 PySpark務以使用熊貓等熱門資料科學 Python 程式庫 NumPy，PyArrow而且無需任何其他設定。

下列範例說明如何封裝 PySpark工作的每個 Python 程式庫。

NumPy

NumPy 是一個用於科學計算的 Python 庫，提供數學，排序，隨機模擬和基本統計的多維數組和運算。若要使用 NumPy，請執行下列命令：

```
import numpy
```

pandas

熊貓是一個建立在之上的 NumPy Python 庫。熊貓庫為數據科學家提供DataFrame數據結構和數據分析工具。要使用熊貓，請運行以下命令：

```
import pandas
```

PyArrow

PyArrow 是一個 Python 庫，用於管理內存中的單欄數據以提高工作性能。 PyArrow 以 Apache Arrow 跨語言開發規範為基礎，這是以單欄格式表示和交換資料的標準方式。若要使用 PyArrow，請執行下列命令：

```
import pyarrow
```

搭配EMR無伺服器使用不同的 Python 版本

除了中的使用案例之外搭配EMR無伺服器使用 Python 程式庫，您還可以使用 Python 虛擬環境使用與 Amazon EMR 無伺服器應用程式在 Amazon 版本中封裝的EMR版本不同的 Python 版本。為此，您必須使用要使用的 Python 版本構建一個 Python 虛擬環境。

若要從 Python 虛擬環境提交工作

1. 使用下列範例中的指令建置虛擬環境。本範例會將 Python 3.9.9 安裝到虛擬環境套件中，並將存檔複製到 Amazon S3 位置。

> **Important**
>
> 如果您使用 Amazon 7.0.0 及更高EMR版本，則必須在 Amazon Linux 2023 環境中執行命令，類似於您用於EMR無伺服器應用程式的指令。
> 如果您使用 6.15.0 或更低版本，則必須在類似的 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 && \
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 to venv
cp -r /usr/local/lib/python3.9/* ./pyspark_venv_python_3.9.9/lib/python3.9/
```

```
# 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，並提交星火工作。

```
# note that the archive suffix "environment" is the same as the directory where you
 copied the Python binary.
--conf spark.archives=s3://DOC-EXAMPLE-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
```

有關如何使用 Python 虛擬環境進行 PySpark 作業的更多信息，請參閱使用虛擬環境。如需如何提交 Spark 工作的更多範例，請參閱Spark 任務。

OSS搭配EMR無伺服器使用三角洲湖

Amazon 6.9.0 及更高EMR版本

> **Note**
>
> Amazon EMR 7.0.0 及更高版本使用三角洲湖 3.0.0，它將`delta-core.jar`文件重命名為。`delta-spark.jar`如果您使用 Amazon EMR 7.0.0 或更高版本，請務必`delta-spark.jar`在您的組態中指定。

Amazon EMR 6.9.0 及更高版本包括 Delta Lake，因此您不再需要自行包裝三角洲湖，也不需要為您的EMR無伺服器任務提供--packages旗標。

1. 當您提交EMR無伺服器工作時，請確定您具有下列組態特性，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. 創建一個本地delta_sample.py來測試創建和讀取 Delta 表。

```
# delta_sample.py
    from pyspark.sql import SparkSession

    import uuid

    url = "s3://DOC-EXAMPLE-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檔案上傳到您的 Amazon S3 儲存貯體。然後使用命start-job-run令將工作送出至現有的EMR無伺服器應用程式。

```
aws s3 cp delta_sample.py s3://DOC-EXAMPLE-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://DOC-EXAMPLE-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，您可以將程式`delta-core`庫封裝為相依性，或將其用作自訂映像檔來新增程式庫。

或者，您可以使`SparkContext.addPyFile`用從`delta-core`JAR檔案中新增 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 6.8.0 及更低EMR版本

如果您使用的是 Amazon EMR 6.8.0 或更低版本，請按照以下步驟將 Delta 湖OSS與EMR無伺服器應用程式搭配使用。

1. 若要建立與 Amazon EMR 無伺服器應用程式上 Spark 版本相容的 Delta Lake 開放原始碼版本，請導覽至三角洲 GitHub並遵循指示進行。
2. 將三角洲湖程式庫上傳到您的 Amazon S3 儲存貯體 AWS 帳戶.
3. 當您在應用程式組態中提交EMR無伺服器工作時，請將目前位於值區中的 Delta Lake JAR 檔案納入。

   ```
   --conf spark.jars=s3://DOC-EXAMPLE-BUCKET/jars/delta-core_2.12-1.1.0.jar
   ```
4. 若要確保您可以讀取和寫入 Delta 資料表，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://DOC-EXAMPLE-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 提交無EMR伺服器任務

Apache Airflow 中的 Amazon Provider 提供無EMR伺服器運算子。如需運算子的詳細資訊，請參閱 Apache Airflow 文件中的 Amazon EMR Serverless 運算子。

您可以使用 EmrServerlessCreateApplicationOperator來建立 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。

若要將 運算子與 Amazon Managed Workflows for Apache Airflow （ MWAA ） 搭配使用，搭配 Airflow 2.2.2，請將以下行新增至您的requirements.txt檔案，並更新您的MWAA環境以使用新檔案。

```
apache-airflow-providers-amazon==6.0.0
boto3>=1.23.9
```

請注意，Amazon 供應商的 5.0.0 版已新增無EMR伺服器支援。6.0.0 版是與 Airflow 2.2.2 相容的最新版本。您可以在 上使用更新版本搭配 Airflow 2.4.3。 MWAA

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 儲存庫中提供完整範例。如需sparkSubmit組態的其他詳細資訊，請參閱 Spark 任務。

```
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無伺服器使用 Hive 使用者定義

Hive 使用者定義函數 (UDFs) 可讓您建立自訂函數來處理記錄或記錄群組。在本教學中，您將使用範例UDF搭配預先存在的 Amazon EMR 無伺服器應用程式來執行輸出查詢結果的任務。若要瞭解如何設定應用程式，請參閱Amazon EMR Serverless 入門。

UDF搭配EMR無伺服器使用

1. 導覽至以GitHub取得範例UDF。克隆回購並切換到您要使用的 git 分支。將存放庫`pom.xml`檔案`maven-compiler-plugin`中的更新為具有來源。同時將目標 java 版本配置更新為`1.8`. 執行`mvn package -DskipTests`以建立包含範例的JAR檔案UDFs。
2. 建立JAR檔案後，請使用下列指令將檔案上傳到 S3 儲存貯體。

   ```
   aws s3 cp brickhouse-0.8.2-JS.jar s3://DOC-EXAMPLE-BUCKET/jars/
   ```

3. 建立範例檔案以使用其中一個範例UDF函數。將此查詢另存為`udf_example.q`並將其上傳到您的 S3 儲存貯體。

   ```
   add jar s3://DOC-EXAMPLE-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. 提交以下蜂巢工作。

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

```
  --application-id application-id \
  --execution-role-arn job-role-arn \
  --job-driver '{
    "hive": {
        "query": "s3://DOC-EXAMPLE-BUCKET/queries/udf_example.q",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://DOC-EXAMPLE-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://DOC-EXAMPLE-BUCKET/logs/"
        }
    }
}'
```

5. 使用get-job-run指令檢查工作的狀態。等待狀態變更為SUCCESS。

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

6. 使用以下命令下載輸出文件。

```
aws s3 cp --recursive s3://DOC-EXAMPLE-BUCKET/logs/applications/application-id/
jobs/job-id/HIVE_DRIVER/ .
```

檔stdout.gz案類似下列。

```
{"key1":[0,1,2],"key2":[3,4,5,6],"key3":[7,8,9]}
2
```

搭配EMR無伺服器使用自訂映像

主題

使用自訂的 Python 版本

您可以建立自訂映像檔以使用不同版本的 Python。例如，若要將 Python 版本 3.10 用於星火工作，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 will run the image as hadoop
USER hadoop:hadoop
```

在您提交 Spark 工作之前，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
```

使用自訂的 Java 版本

下面的例子演示了如何構建一個自定義映像使用 Java 11 為您的 Spark 作業。

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

USER root

# install JDK 11
```

```
RUN sudo amazon-linux-extras install java-openjdk11

# EMRS will run the image as hadoop
USER hadoop:hadoop
```

提交星火作業之前，設置星火屬性使用 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-
```

建立資料科學影像

下面的例子演示了如何包括常見的，數據科學 Python 包，如熊貓和 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 will run the image as hadoop
USER hadoop:hadoop
```

使用 Apache 塞多納處理地理空間資料

下面的示例演示了如何構建一個圖像，以包括 Apache Sedona 進行地理空間處理。

```
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 will run the image as hadoop
USER hadoop:hadoop
```

在 Amazon 無服務器上使用亞馬遜紅移集成阿帕奇星火 EMR

在 Amazon 6.9.0 及更高EMR版本中，每個版本映像都包含 Apache 星火和 Amazon Redshift 之間的連接器。有了這個連接器，您可以使用亞馬遜EMR無伺服器上的 Spark 來處理儲存在 Amazon Redshift 中的資料。整合是以 `spark-redshift` 開放原始碼連接器為基礎。對於 Amazon EMR 無服務器，阿帕奇星火的 Amazon Redshift 集成作為原生集成包括在內。

主題

- 啟動一個星火應用程序與 Amazon Redshift 集成阿帕奇星火
- 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

啟動一個星火應用程序與 Amazon Redshift 集成阿帕奇星火

若要使用與EMR無伺服器 6.9.0 的整合，您必須將必要的 Spark Redshift 相依性與 Spark 工作傳遞。用`--jars`於包括與 Redshift 連接器相關的程式庫。若要查看 `--jars` 選項支援的其他檔案位置，請參閱 Apache Spark 說明文件的進階相依性管理一節。

- `spark-redshift.jar`
- `spark-avro.jar`
- `RedshiftJDBC.jar`
- `minimal-json.jar`

Amazon 6.10.0 及更高EMR版本不需要相`minimal-json.jar`依性，預設會自動將其他相依性安裝到每個叢集。下列範例說明如何為 Apache Spark 啟動與 Amazon Redshift 整合的 Spark 應用程式。

Amazon EMR 6.10.0 +

在EMR無伺服器版本 6.10.0 及更高版本上，利用 Amazon Redshift 整合，在亞馬遜EMR無伺服器上啟動 Spark 任務。

```
spark-submit my_script.py
```

Amazon EMR 6.9.0

若要在 Amazon EMR 無伺服器上使用 Amazon Redshift 整合在無伺服EMR器 6.9.0 版本上啟動 Spark 任務，請使用下列範例所示的`--jars`選項。請注意，與`--jars`選項一起列示的路徑是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 檢索憑據並連接到 Amazon Redshift

您可以將登入資料儲存在 Secrets Manager 中，以安全地向 Amazon Redshift 進行身份驗證，並讓 Spark 任務呼叫`GetSecretValue`API來擷取它：

```
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
```

使用驅動程式向 Amazon Redshift 進行驗證 JDBC

設置用戶名和密碼 JDBC URL

您可以在中指定 Amazon Redshift 資料庫名稱和密碼，以向 Amazon Redshift 叢集驗證星火任務。JDBC URL

> **Note**
>
> 如果您在中傳遞資料庫認證URL，則擁有存取權的任何人也URL可以存取認證。通常不建議使用此方法，因為它不安全。

如果您的應用程式無法考慮安全性，您可以使用下列格式在中設定使用者名稱和密碼 JDBCURL：

```
jdbc:redshift://redshifthost:5439/database?user=username&password=password
```

搭配 Amazon EMR 無伺服器任務執行角色使用IAM基於身份驗證

從 Amazon EMR 無伺服器版本 6.9.0 開始，Amazon Redshift JDBC 驅動程式 2.1 或更高版本已封裝到環境中。使用 2.1 及更高版本的JDBC驅動程式，您可以指定JDBCURL且不包含原始使用者名稱和密碼。

相反地，可以指定 `jdbc:redshift:iam://` 配置。這會命令JDBC驅動程式使用您的EMR無伺服器作業執行角色來自動擷取認證。如需詳細資訊，請參閱 Amazon Redshift 管理指南中的設定JDBC或ODBC連線以使用IAM登入資料。這方面的一個例子URL是：

```
jdbc:redshift:iam://examplecluster.abc123xyz789.us-west-2.redshift.amazonaws.com:5439/
dev
```

符合提供的條件時，您的工作執行角色需要下列權限：

權限	作業執行角色所需的條件
`redshift:GetClusterCredentials`	JDBC驅動程序需要從 Amazon Redshift 獲取憑據
`redshift:DescribeCluster`	如果您指定了 Amazon Redshift 叢集和 AWS 區域 在JDBCURL而不是端點
`redshift-serverless:GetCredentials`	JDBC驅動程式需要從 Amazon Redshift 無伺服器擷取登入資料
`redshift-serverless:GetWorkgroup`	如果您使用的是 Amazon Redshift 無伺服器，且您要根據工作群組名稱和URL區域指定

在不同的地方連接到 Amazon Redshift VPC

當您在下方設定佈建的 Amazon Redshift 叢集或 Amazon Redshift 無伺服器工作群組時VPC，必須為 Amazon EMR 無伺服器應用程式設定VPC連線，才能存取資源。如需如何在EMR無伺服器應用程式上設定VPC連線的詳細資訊，請參閱設定VPC存取權。

- 如果您佈建的 Amazon Redshift 叢集或 Amazon Redshift 無伺服器工作群組可公開存取，您可以指定在建立無伺服器應用程式時已連接NAT閘道的一或多個私有子網路。EMR
- 如果您佈建的 Amazon Redshift 叢集或 Amazon Redshift 無伺服器工作群組無法公開存取，您必須依照中所述，為您的 Amazon Redshift 叢集建立 Amazon Redshift 受管VPC端點。設定VPC存取權或者，您也可以按照亞馬遜 Redshift 管理指南中的連接到 Amazon Redshift 無伺服器中的說明來建立自己的 Amazon Redshift 無伺服器工作群組。您必須將叢集或子群組與建立EMR無伺服器應用程式時指定的私人子網路建立關聯。

> **Note**
>
> 如果您使用IAM基於身份驗證，且EMR無伺服器應用程式的私有子網路沒有連接NAT閘道，則還必須在這些子網路上建立適用於 Amazon Redshift 或 Amazon Redshift 無伺服器的VPC端點。這樣，JDBC驅動程序可以獲取憑據。

在 Amazon Redshift 中讀取和寫入

下列程式碼範例用 PySpark 於從具有資料來源和 Spark SQL 的 Amazon Redshift 資料庫讀取API和寫入範例資料。

Data source API

用 PySpark 於從具有資料來源的 Amazon Redshift 資料庫讀取和寫入範例資料API。

```
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

用於 PySpark 使用 Spark SQL 從 Amazon Redshift 資料庫讀取和寫入範例資料。

```
import boto3
import json
import sys
import os
```

```
from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .enableHiveSupport() \
    .getOrCreate()

url = "jdbc:redshift:iam://redshifthost:5439/database"
aws_iam_role_arn = "arn:aws:iam::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

- 我們建議您打開從 Amazon 上SSL的星火EMR到亞馬 Amazon Redshift 的JDBC連接。
- 我們建議您在中管理 Amazon Redshift 叢集的登入資料 AWS Secrets Manager 作為最佳實踐。請參閱使用 AWS Secrets Manager 檢索用於連接到 Amazon Redshift 的憑據舉個例子。
- 我們建議您傳遞具有 Amazon Redshift 身份驗證`aws_iam_role`參數參數的IAM角色。
- 參數 `tempformat` 目前不支援 Parquet 格式。
- 這些`tempdir`URI指向 Amazon S3 位置。此暫時目錄不會自動清理，因此可能會增加額外的費用。
- 請考慮下列針對 Amazon Redshift 的建議：
  - 建議您封鎖對 Amazon Redshift 叢集的公開存取。

  - 建議您開啟 Amazon Redshift 稽核日誌。
  - 建議您開啟 Amazon Redshift 靜態加密。
- 請考慮下列針對 Amazon S3 的建議：
  - 建議您封鎖對 Amazon S3 儲存貯體的公開存取。
  - 建議您使用 Amazon S3 伺服器端加密來加密所用的 S3 儲存貯體。
  - 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：
    - 設定 AWS `hadoop-env`組態分類中的存取金鑰
    - 編碼 AWS 存取金鑰 `tempdir` URI

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 將資料複製到您可以查詢的 Amazon DynamoDB 資料表。

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 中的建立儲存貯體中的指示。將 的參考取代*`amzn-s3-demo-bucket`*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. 從封存中擷取`features.txt`檔案，並檢視檔案中的前幾行：

   ```
   unzip features.zip
   ```

```
head features.txt
```

結果看起來應該類似於以下內容。

```
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
```

此處各行中的欄位表示唯一識別符、名稱、自然特徵類型、狀態、緯度、經度和英尺高度。

3. 將資料上傳至 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://DOC-EXAMPLE_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

若要從您在上一個步驟中建立的 Hive 資料表將資料複製到 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，您可以從文字檔案直接讀取資料，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

若要查詢您在上一個步驟中建立的 DynamoDB 資料表中的資料，您可以使用 Spark SQL或 Spark MapReduce API。

Example – 使用 Spark 查詢 DynamoDB 資料表 SQL

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 查詢 DynamoDB 資料表 MapReduce API

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

若要查詢您在上一個步驟中建立的 DynamoDB 資料表中的資料，請遵循查詢 DynamoDB 資料表中的資料中的指示。

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，AccountB也是 Amazon DynamoDB 所在的帳戶。

1. 在 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 中建立AccountB可存取 DynamoDB 資料表Cross-Account-Role-BIAM的角色。

   a. 登入 AWS Management Console 並在 開啟IAM主控台https://console.aws.amazon.com/iam/。

   b. 選擇角色 ，然後建立新的名為 的角色Cross-Account-Role-B。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政策連接至 Cross-Account-Role-B。

      以下是授予 DynamoDB 資料表 存取權的政策CrossAccountTable。

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

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

e. 授予 以擔任 Job-Execution-Role-AAccountA的- STS Assume role許可Cross-Account-Role-B。

在 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。

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

f. 設定值為 的 dynamodb.customAWSCredentialsProvider 屬性，如同核心站台分類com.amazonaws.emr.AssumeRoleAWSCredentialsProvider。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

搭配 Apache Spark 使用 DynamoDB 連接器時的考量

- 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。
- 若要將查詢轉換為 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。述詞下推功能只會在搭配 Spark 使用連接器時運作SQL，而不適用於 時運作 MapReduce API。

搭配 Apache Hive 使用 DynamoDB 連接器時的考量

調整映射器數量上限

- 如果您使用`SELECT`查詢從對應至 DynamoDB 的外部 Hive 資料表讀取資料，EMR則 Serverless 上的地圖任務數量會計算為為為 DynamoDB 資料表設定的讀取輸送量總計，除以每個地圖任務的輸送量。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無EMR伺服器應用程式設定的容器數量上限的映射任務數量，具體取決於為 DynamoDB 設定的讀取輸送量。此外，長時間執行的 Hive 查詢可能會消耗 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地圖任務讀取的資料量。
- 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 任務在從 DynamoDB 資料表讀取`dynamodb.max.map.tasks`時最多會使用 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能會消耗所有寫入輸送量 。這會導致對相同任務或其他任務中其他任務的寫入進行限流。

- 為了避免寫入限流，您可以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來設定 `mapreduce.job.maps` 屬性的值。在無EMR伺服器start-job-run命令的`mapred-site`分類中設定此屬性。

安全

雲端安全 AWS 是最高的優先級。作為 AWS 客戶，您可以從資料中心和網路架構中獲益，該架構專為滿足對安全性最敏感的組織的需求而打造。

安全是兩者之間的共同責任 AWS 和你。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 負責保護運行的基礎設施 AWS 中的服務 AWS 雲端。 AWS 還為您提供可以安全使用的服務。第三方稽核員會定期測試和驗證我們安全性的有效性，作為 AWS 合規方案。若要了解適用於 Amazon EMR 無伺服器的合規計劃，請參閱 AWS 合規計劃範圍內的服務。
- 雲端中的安全性 — 您的責任取決於 AWS 您使用的服務。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 無伺服器時套用共同責任模型。本章的主題說明如何設定 Amazon EMR 無伺服器以及如何使用其他 AWS 符合您安全性與合規目標的服務。

主題

Amazon EMR 無伺服器的安全性最佳實務

Amazon EMR 無伺服器提供許多安全功能，可在您開發和實作自己的安全政策時考慮。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無伺服器會為使用角色 (例如執行IAM角色) 的應用程式提供精細的存取原則。建議只授予作業所需的最低權限集，例如覆蓋應用程式和日誌目的地的存取權限。我們還建議定期以及在應用程式碼發生變更時審核作業許可。

隔離不受信任的應用程式碼

EMR無伺服器會在屬於不同無伺服器應用EMR程式的工作間建立完整的網路隔離 在需要工作層級隔離的情況下，請考慮將工作隔離到不同EMR的無伺服器應用程式中。

以角色為基礎的存取控制 (RBAC) 權限

系統管理員應嚴格控制EMR無伺服器應用程式的角色型存取控制 (RBAC) 權限。

資料保護

AWS 共同責任模型適用於 Amazon EMR Serverless 中的資料保護。如本模型所述， AWS 負責保護執行所有 AWS Cloud 的全域基礎設施。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。 FAQ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。

為了資料保護目的，我們建議您保護 AWS 帳戶憑證，並使用 AWS Identity and Access Management （ ）設定個別帳戶IAM。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：

- 對每個帳戶使用多重要素驗證 （ 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 和使用者活動日誌 AWS CloudTrail。
- 使用 AWS 加密解決方案，以及 服務中的所有 AWS 預設安全控制項。
- 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 透過命令列介面或 FIPS 存取 時需要 140-2 個經過驗證的密碼編譯模組API，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客戶帳戶號碼等敏感的識別資訊，放在自由格式的欄位中，例如Name (名稱) 欄位。這包括當您使用主控台、 或 使用 Amazon EMR Serverless API AWS CLI或其他 AWS 服務時 AWS SDKs。您輸入 Amazon EMR Serverless 或其他 服務的任何資料都可能被擷取以包含在診斷日誌中。當您將 URL提供給外部伺服器時，請勿在 中包含憑證資訊，URL以驗證您對該伺服器的請求。

靜態加密

資料加密有助於防止未經授權的使用者讀取叢集上的資料和相關的資料儲存體系統。這包括儲存到持久性媒體的資料 (稱為靜態資料)，以及透過網路傳送時可能會被攔截的資料 (稱為傳輸中資料)。

資料加密需要金鑰和憑證。您可以從多個選項中選擇，包括 管理的金鑰 AWS Key Management Service、Amazon S3 管理的金鑰，以及來自您提供的自訂提供者的金鑰和憑證。使用 AWS KMS 作為金鑰提供者時，儲存和使用加密金鑰會收取費用。如需詳細資訊，請參閱 AWS KMS 定價。

在指定加密選項之前，請先決定要使用的金鑰和憑證管理系統。然後為您指定作為加密設定一部分的自訂提供者建立金鑰和憑證。

Amazon S3 中EMRFS資料靜態加密

每個 EMR Serverless 應用程式使用特定的版本，其中包括 EMRFS（EMR 檔案系統）。Amazon S3 加密適用於從 EMR Amazon S3 讀取和寫入的檔案系統（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，您可以使用 Per bucket encryption overrides (每個儲存貯體加密覆寫) 為個別儲存貯體指定不同的加密方法。無論是否啟用 Amazon S3 加密，Transport Layer Security（TLS）都會加密EMR叢集節點和 Amazon S3 之間傳輸中的EMRFS物件。如果您將 Amazon S3 CSE與客戶受管金鑰搭配使用，則用於在無EMR伺服器應用程式中執行任務的執行角色必須具有金鑰的存取權。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，儲存和使用加密金鑰會收取費用。如需詳細資訊，請參閱 AWS KMS 定價。

Amazon S3 伺服器端加密

當您設定 Amazon S3 伺服器端加密時，Amazon S3 會在將資料寫入磁碟時在物件層級加密資料，並在存取時解密資料。如需 的詳細資訊SSE，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 SSE中指定 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 會為您管理金鑰。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。

若要對寫入 Amazon S3 的資料使用 AWS KMS 加密，當您使用 `StartJobRun` 時，有兩個選項API。您可以為寫入 Amazon S3 的所有內容啟用加密，也可以為寫入特定儲存貯體的資料啟用加密。如需 `StartJobRun` 的詳細資訊API，請參閱EMR無伺服器API參考 。

若要開啟寫入 Amazon S3 的所有資料的 AWS KMS 加密，請在呼叫 `StartJobRun` 時使用下列命令API。

```
--conf spark.hadoop.fs.s3.enableServerSideEncryption=true
--conf spark.hadoop.fs.s3.serverSideEncryption.kms.keyId=<kms_id>
```

若要開啟寫入特定儲存貯體的資料 AWS KMS 加密，請在呼叫 `StartJobRun` 時使用下列命令API。

```
--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>
```

SSE 搭配客戶提供的金鑰（SSE-C）無法使用 EMR Serverless。

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。- 和 CSE-KMSCSEC 的加密詳情略有不同，具體取決於指定的提供者和要解密或加密的物件中繼資料。如果您將 Amazon S3 CSE與客戶受管金鑰搭配使用，則用於在無EMR伺服器應用程式中執行任務的執行角色必須具有金鑰的存取權。可能會KMS收取額外費用。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

儲存在暫時儲存中的資料會使用業界標準的 AES-256 密碼編譯演算法，使用服務擁有的金鑰加密。

金鑰管理

您可以設定 KMS 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，請參閱輪換客戶主金鑰 。

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 驅動程式與執行程式之間的通訊會經過身分驗證且為內部通訊。RPC 驅動程式與執行程式之間的通訊會加密。
- Hive
  - AWS Glue 中繼存放區與無EMR伺服器應用程式之間的通訊會透過 進行TLS。

您應該只允許使用 Amazon S3 儲存貯體IAM政策上的 aws：SecureTransport condition 對 HTTPS（TLS）進行加密連線。

Amazon EMR Serverless 中的身分和存取管理（IAM）

AWS Identity and Access Management（IAM）是一種 AWS 服務 ，可協助管理員安全地控制對 AWS 資源的存取。IAM 管理員會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 AWS 服務 您可以免費使用的 。

主題

- 物件
- 使用身分驗證
- 使用政策管理存取權
- EMR Serverless 如何使用 IAM

- 針EMR對無伺服器使用服務連結角色
- Amazon EMR Serverless 的任務執行期角色
- EMR Serverless 的使用者存取政策範例
- 標籤型存取控制的政策
- EMR Serverless 的身分型政策範例
- 受 AWS 管政策的 Amazon EMR Serverless 更新
- 對 Amazon EMR Serverless 身分和存取進行故障診斷

物件

使用 AWS Identity and Access Management （ IAM ） 的方式會有所不同，取決於您在 Amazon EMR Serverless 中執行的工作。

服務使用者 – 如果您使用 Amazon EMR Serverless 服務來執行您的工作，您的管理員會為您提供所需的憑證和許可。當您使用更多 Amazon EMR Serverless 功能來執行工作時，您可能需要額外的許可。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法存取 Amazon EMR Serverless 中的功能，請參閱 對 Amazon EMR Serverless 身分和存取進行故障診斷。

服務管理員 – 如果您在公司負責 Amazon EMR Serverless 資源，您可能可以完整存取 Amazon EMR Serverless。您的任務是判斷您的服務使用者應該存取哪些 Amazon EMR Serverless 功能和資源。然後，您必須向IAM管理員提交請求，以變更服務使用者的許可。請檢閱此頁面上的資訊，以了解 的基本概念IAM。若要進一步了解貴公司如何IAM搭配 Amazon EMR Serverless 使用，請參閱 Amazon EMR Serverless 中的身分和存取管理 （ IAM ）。

IAM 管理員 – 如果您是IAM管理員，您可能想要了解如何撰寫政策以管理 Amazon EMR Serverless 存取的詳細資訊。若要檢視您可以在 中使用的 Amazon EMR Serverless 身分型政策範例IAM，請參閱 EMR Serverless 的身分型政策範例。

使用身分驗證

驗證是您 AWS 使用身分憑證登入 的方式。您必須以身分 AWS 帳戶根使用者、IAM使用者身分或擔任IAM角色來驗證 （ 登入 AWS ）。

您可以使用透過身分來源提供的憑證，以聯合身分 AWS 身分登入 。 AWS IAM Identity Center （ IAM Identity Center ） 使用者、您公司的單一登入身分驗證，以及您的 Google 或 Facebook 憑證，都是聯合身分的範例。當您以聯合身分登入時，您的管理員先前會使用 IAM角色設定身分聯合。當您 AWS 使用聯合存取 時，您間接擔任 角色。

您可以登入 AWS Management Console 或 AWS 存取入口網站，視您是的使用者類型而定。如需登入的詳細資訊 AWS，請參閱 使用者指南 中的如何登入 AWS 帳戶您的 。 AWS 登入

如果您以 AWS 程式設計方式存取 ， AWS 會提供軟體開發套件 （SDK） 和命令列介面 （CLI），以使用您的憑證以密碼編譯方式簽署您的請求。如果您不使用 AWS 工具，則必須自行簽署請求。如需使用建議方法自行簽署請求的詳細資訊，請參閱 IAM 使用者指南 中的簽署 AWS API請求。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素驗證 （MFA） 來提高帳戶的安全性。若要進一步了解，請參閱 AWS IAM Identity Center 使用者指南中的多重要素驗證，以及 IAM 使用者指南 中的使用多重要素驗證 （MFA） AWS。

AWS 帳戶 根使用者

當您建立 時 AWS 帳戶，您會從一個登入身分開始，該身分可完全存取 帳戶中的所有 AWS 服務 和資源。此身分稱為 AWS 帳戶 根使用者，透過您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需需要您以根使用者身分登入的任務完整清單，請參閱 IAM 使用者指南 中的需要根使用者憑證的任務。

聯合身分

最佳實務是， 要求人類使用者，包括需要管理員存取權的使用者，使用 AWS 服務 臨時憑證與身分提供者聯合來存取 。

聯合身分是來自您的企業使用者目錄、Web 身分提供者、 AWS Directory Service、身分中心目錄，或使用透過身分來源提供的 AWS 服務 憑證存取的任何使用者。當聯合身分存取 時 AWS 帳戶，它們會擔任 角色，而角色會提供臨時憑證。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，或者您可以連線並同步到您身分來源中的一組使用者 AWS 帳戶 和群組，以便在所有 和應用程式中使用。如需 IAM Identity Center 的相關資訊，請參閱 AWS IAM Identity Center 使用者指南 中的什麼是 IAM Identity Center？。

IAM 使用者和群組

IAM 使用者是 中具有單一個人或應用程式特定許可 AWS 帳戶 的身分。如果可能，我們建議您依賴臨時憑證，而不是建立具有密碼和存取金鑰等長期憑證IAM的使用者。不過，如果您有特定的使用案例需要IAM使用者長期憑證，建議您輪換存取金鑰。如需詳細資訊，請參閱 IAM 使用者指南 中的針對需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是指定IAM使用者集合的身分。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有名為 的群組IAMAdmins，並授予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。若要進一步了解，請參閱 IAM 使用者指南 中的何時建立IAM使用者 （而非角色）。

IAM 角色

IAM 角色是 中具有特定許可 AWS 帳戶 的身分。它類似於IAM使用者，但與特定人員無關。您可以透過 AWS Management Console 切換IAM角色 暫時在 中擔任角色。 https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-console.html您可以透過呼叫 AWS CLI 或 AWS API 操作，或使用自訂 來擔任角色URL。如需使用角色方法的詳細資訊，請參閱 IAM 使用者指南 中的擔任角色的方法。

IAM 具有臨時憑證的角色在下列情況下很有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需聯合角色的相關資訊，請參閱 IAM 使用者指南 中的為第三方身分提供者建立角色。如果您使用 IAM Identity Center，您可以設定許可集。若要控制身分在身分驗證後可以存取的內容，IAMIdentity Center 會將許可集與 中的角色相關聯IAM。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。
- 臨時IAM使用者許可 – IAM使用者或角色可以擔任IAM角色，暫時接受特定任務的不同許可。
- 跨帳戶存取 – 您可以使用 IAM角色，允許不同帳戶中的某人 （受信任的主體） 存取您帳戶中的資源。角色是授予跨帳戶存取權的主要方式。不過，使用某些 AWS 服務，您可以將政策直接連接至資源 （而不是使用角色作為代理）。若要了解跨帳戶存取的角色和資源型政策之間的差異，請參閱 IAM 使用者指南 中的跨帳戶資源存取IAM。
- 跨服務存取 – 有些 AWS 服務 使用其他 中的功能 AWS 服務。例如，當您在 服務中撥打電話時，該服務通常會在 Amazon 中執行應用程式EC2或在 Amazon S3 中儲存物件。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉送存取工作階段 （FAS） – 當您使用IAM使用者或角色在 中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 使用呼叫 的委託人許可 AWS 服務，並結合請求向下游服務 AWS 服務 提出請求。FAS 只有在服務收到需要與其他 AWS 服務 或 資源互動才能完成的請求時，才會發出請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出FAS請求的政策詳細資訊，請參閱轉送存取工作階段 。

  - 服務角色 – 服務角色是服務代表您執行動作時擔任IAM的角色。IAM 管理員可以從 內部建立、修改和刪除服務角色IAM。如需詳細資訊，請參閱 使用者指南 中的建立角色以將許可委派給 AWS 服務 。 IAM
  - 服務連結角色 – 服務連結角色是連結至 的服務角色類型 AWS 服務。服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的 中 AWS 帳戶 ，並由 服務擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon 上執行的應用程式 EC2 – 您可以使用 IAM角色來管理在EC2執行個體上執行之應用程式的臨時憑證，以及提出 AWS CLI 或 AWS API請求。最好將存取金鑰存放在EC2執行個體中。若要將 AWS 角色指派給EC2執行個體並將其提供給其所有應用程式，您可以建立連接至執行個體的執行個體設定檔。執行個體設定檔包含 角色，並啟用在EC2執行個體上執行的程式，以取得臨時憑證。如需詳細資訊，請參閱 IAM 使用者指南 中的使用 IAM角色將許可授予在 Amazon EC2執行個體上執行的應用程式。

若要了解如何使用IAM角色或IAM使用者，請參閱 IAM 使用者指南 中的建立IAM角色 （ 而非使用者 ）的時機。

使用政策管理存取權

您可以透過建立政策並將其連接至 AWS 身分或資源 AWS 來控制 中的存取。政策是 AWS 其中的物件，當與身分或資源相關聯時， 會定義其許可。當主體 （ 使用者、根使用者或角色工作階段 ） 發出請求時， 會 AWS 評估這些政策。政策中的許可決定是否允許或拒絕請求。大多數政策會以JSON文件 AWS 形式儲存在 中。如需JSON政策文件結構和內容的詳細資訊，請參閱 IAM 使用者指南 中的JSON政策概觀。

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

預設情況下，使用者和角色沒有許可。若要授予使用者對所需資源執行動作的許可，IAM管理員可以建立IAM政策。然後，管理員可以將IAM政策新增至角色，使用者可以擔任角色。

IAM 無論您用來執行操作的方法為何，政策都會定義動作的許可。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該政策的使用者可以從 AWS Management Console、 AWS CLI或 AWS 取得角色資訊API。

身分型政策

身分型政策是您可以連接到身分的JSON許可政策文件，例如IAM使用者、使用者群組或角色。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分型政策，請參閱 IAM 使用者指南 中的建立IAM政策。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管政策是獨立的政策，您可以連接到 中的多個使用者、群組和角色 AWS 帳戶。受管政策包括 AWS 受管政策和客戶受管政策。若要了解如何在受管政策或內嵌政策之間進行選擇，請參閱 IAM 使用者指南 中的在受管政策與內嵌政策之間進行選擇。

資源型政策

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

資源型政策是位於該服務中的內嵌政策。您無法在資源型政策IAM中使用來自 的 AWS 受管政策。

存取控制清單 （ ACLs ）

存取控制清單 （ ACLs ） 控制哪些主體 （ 帳戶成員、使用者或角色 ） 具有存取 資源的許可。ACLs 類似於資源型政策，雖然它們不使用JSON政策文件格式。

Amazon S3 AWS WAF和 Amazon VPC是支援 的服務範例ACLs。若要進一步了解 ACLs，請參閱 Amazon Simple Storage Service 開發人員指南 中的存取控制清單 （ ACL ） 概觀。

其他政策類型

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

- 許可界限 – 許可界限是一項進階功能，您可以在其中設定身分型政策可授予IAM實體 （ IAM使用者或角色 ） 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可界限的詳細資訊，請參閱 IAM 使用者指南 中的IAM實體許可界限。
- 服務控制政策 （ SCPs ） – SCPs是在 中指定組織或組織單位 （ OU ） 最大許可JSON的政策 AWS Organizations。 AWS Organizations 是一項用於分組和集中管理您企業 AWS 帳戶 所擁有多個項目的服務。如果您啟用組織中的所有功能，則可以將服務控制政策 （ SCPs ） 套用至任何或所有帳

戶。SCP 限制成員帳戶中實體的許可，包括每個 AWS 帳戶根使用者。如需 Organizations 和 的詳細資訊SCPs，請參閱 AWS Organizations 使用者指南 中的服務控制政策。

- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱 IAM 使用者指南 中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。若要了解如何 AWS 在涉及多種政策類型時決定是否允許請求，請參閱 IAM 使用者指南 中的政策評估邏輯。

EMR Serverless 如何使用 IAM

在您使用 IAM 管理 Amazon EMR Serverless 的存取權之前，請先了解哪些IAM功能可用於 Amazon EMR Serverless。

IAM 您可以與 EMR Serverless 搭配使用的功能

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
ACLs	否
ABAC （政策中的標籤）	是
暫時性憑證	是
主體許可	是
服務角色	否

IAM 功能	Amazon EMR Serverless 支援
服務連結角色	是

若要深入了解 EMR Serverless 和其他 AWS 服務如何搭配大多數IAM功能使用，請參閱 IAM 使用者指南 中的 AWS 服務IAM。

EMR Serverless 的身分型政策

支援身分型政策：是

身分型政策是JSON許可政策文件，您可以附加到身分，例如IAM使用者、使用者群組或角色。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分型政策，請參閱 IAM 使用者指南 中的建立IAM政策。

透過身分IAM型政策，您可以指定允許或拒絕的動作和資源，以及允許或拒絕動作的條件。您無法在身分型政策中指定主體，因為這會套用至連接的使用者或角色。若要了解您可以在JSON政策中使用的所有元素，請參閱 IAM 使用者指南 中的IAMJSON政策元素參考。

EMR Serverless 的身分型政策範例

若要檢視 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

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

若要啟用跨帳戶存取，您可以將另一個帳戶中的整個帳戶或IAM實體指定為資源型政策中的主體。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當主體和資源位於不同的 時 AWS 帳戶，受信任帳戶中的IAM管理員也必須授予主體實體（使用者或角色）存取資源的許可。其透過將身分型政策連接到實體來授與許可。不過，如果資源型政策會為相同帳戶中的主體授予存取，這時就不需要額外的身分型政策。如需詳細資訊，請參閱 IAM 使用者指南 中的跨帳戶資源存取權IAM。

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什麼。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`元素說明您可以用來允許或拒絕政策中存取的動作。政策動作通常具有與相關聯 AWS API操作相同的名稱。有一些例外狀況，例如沒有相符API操作的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

若要查看無EMR伺服器動作清單，請參閱服務授權參考 中的 Amazon EMR Serverless 的動作、資源和條件金鑰。

EMR Serverless 中的政策動作在動作之前使用下列字首。

```
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"
         ]
```

若要檢視 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什麼。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 政策元素會指定動作套用的物件。陳述式必須包含 `Resource` 或 `NotResource` 元素。最佳實務是使用其 Amazon Resource Name （ ARN ） 指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

若要查看 Amazon EMR Serverless 資源類型及其 的清單ARNs，請參閱服務授權參考 中的 Amazon EMR Serverless 定義的資源。若要了解您可以指定每個資源 ARN 的動作，請參閱 Amazon EMR Serverless 的動作、資源和條件索引鍵。

若要檢視 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 的政策條件金鑰

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什麼。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

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

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

您也可以在指定條件時使用預留位置變數。例如，只有在使用者使用其IAM使用者名稱加上標籤時，您才能授予IAM使用者存取資源的許可。如需詳細資訊，請參閱 IAM 使用者指南 中的IAM政策元素：變數和標籤。

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

若要查看 Amazon EMR Serverless 條件索引鍵的清單，並了解您可以使用條件索引鍵的動作和資源，請參閱服務授權參考 中的 Amazon EMR Serverless 的動作、資源和條件索引鍵。

所有 Amazon EC2動作都支援 aws:RequestedRegion和 ec2:Region 條件金鑰。如需詳細資訊，請參閱範例：限制對特定區域 的存取。

Serverless EMR 中的存取控制清單 （ACLs）

支援 ACLs：否

存取控制清單 （ACLs） 控制哪些主體 （帳戶成員、使用者或角色） 具有存取 資源的許可。ACLs 類似於資源型政策，雖然它們不使用JSON政策文件格式。

使用 EMR Serverless 的屬性型存取控制 （ABAC）

支援 ABAC（政策中的標籤）	是

屬性型存取控制 （ABAC） 是一種根據屬性定義許可的授權策略。在 中 AWS，這些屬性稱為標籤。您可以將標籤連接至IAM實體 （使用者或角色） 和許多 AWS 資源。標記實體和資源是 的第一步ABAC。然後，您可以設計ABAC政策，以便在主體的標籤與其嘗試存取之資源上的標籤相符時允許操作。

ABAC 有助於快速成長的環境，並有助於處理政策管理變得繁瑣的情況。

如需根據標籤控制存取，請使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 條件索引鍵，在政策的條件元素中，提供標籤資訊。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需 的詳細資訊ABAC，請參閱 使用者指南 中的什麼是 ABAC？。 IAM 若要檢視包含設定 之步驟的教學課程ABAC，請參閱 IAM 使用者指南 中的使用屬性型存取控制 （ABAC）。

將臨時憑證與 Serverless EMR 搭配使用

支援臨時憑證：是

當您使用臨時憑證登入時，有些 AWS 服務 無法使用。如需其他資訊，包括 AWS 服務 使用哪些臨時憑證，請參閱 IAM 使用者指南 中的 AWS 服務 與 搭配使用IAM。

如果您 AWS Management Console 使用使用者名稱和密碼以外的任何方法登入 ，則表示您正在使用臨時憑證。例如，當您 AWS 使用公司的單一登入 （SSO） 連結存取 時，該程序會自動建立臨時憑證。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需切換角色的詳細資訊，請參閱 IAM 使用者指南 中的切換到角色 （主控台）。

您可以使用 AWS CLI 或 手動建立臨時憑證 AWS API。然後，您可以使用這些臨時登入資料來存取 AWS. AWS recommends，以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 中的臨時安全憑證IAM。

EMR Serverless 的跨服務主體許可

支援轉送存取工作階段 （FAS）：是

當您使用IAM使用者或角色在 中執行動作時 AWS，您會被視為委託人。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 使用呼叫 的委託人許可 AWS 服務，並結合 AWS 服務 請求向下游服務提出請求的 。FAS 只有在服務收到需要與其他 AWS 服務 或 資源互動才能完成的請求時，才會提出請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出FAS請求時的政策詳細資訊，請參閱轉送存取工作階段 。

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 的服務IAM。在表格中尋找服務，其中包含服務連結角色欄中的 Yes。選擇是連結，以檢視該服務的服務連結角色文件。

針EMR對無伺服器使用服務連結角色

Amazon EMR 無伺服器使用 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無伺服器的唯一IAM角色類型。服務連結角色由EMR無伺服器預先定義，並包含服務呼叫其他人所需的所有權限 AWS 代表您提供的服務。

服務連結角色可讓EMR無伺服器的設定更容易，因為您不需要手動新增必要的權限。EMR無伺服器會定義其服務連結角色的權限，除非另有定義，否則只有EMR無伺服器可以擔任其角色。定義的權限包括信任原則和權限原則，而且該權限原則無法附加至任何其他IAM實體。

您必須先刪除服務連結角色的相關資源，才能將其刪除。這樣可以保護您的EMR無伺服器資源，因為您無法意外移除存取資源的權限。

如需支援服務連結角色之其他服務的相關資訊，請參閱 AWS 使用的服務，IAM並在服務連結角色欄中尋找具有 [是] 的服務。選擇具有連結的是，以檢視該服務的服務連結角色文件。

無伺服器EMR的服務連結角色權限

EMR無伺服器會使用名為的服務連結角色，讓其能AWSServiceRoleForAmazonEMRServerless夠呼叫AWS APIs代表您。

服 AWSServiceRoleForAmazonEMRServerless 務連結角色會信任下列服務擔任該角色：

- `ops.emr-serverless.amazonaws.com`

名為的角色權限原則`AmazonEMRServerlessServiceRolePolicy`允許EMR無伺服器對指定的資源完成下列動作。

> **Note**
>
> 受管理的原則內容會變更，因此此處顯示的政策可能已過期。檢視中最多的 up-to-date 原 mazonEMRServerlessServiceRolePolicy則 A AWS Management Console.

- 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。

```
{
    "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"
                    ]
                }
            }
        }
    ]
}
```

下列信任原則已附加至此角色，以允許EMR無伺服器主體擔任此角色。

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

```
                "Service": [
                    "ops.emr-serverless.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

您必須設定權限，才能允許IAM實體 (例如使用者、群組或角色) 建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱IAM使用指南中的服務連結角色權限。

建立無伺服器EMR的服務連結角色

您不需要手動建立一個服務連結角色。當您在 EMR AWS Management Console (使用EMR工作室), AWS CLI，或 AWS API，EMR無伺服器會為您建立服務連結角色。您必須設定權限，才能允許IAM實體 (例如使用者、群組或角色) 建立、編輯或刪除服務連結角色。

若要使用建立 AWSServiceRoleForAmazonEMRServerless 服務連結角色 IAM

針對需要建立服務連結角色的IAM實體，將下列陳述式新增至權限原則。

```
{
    "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"}}
}
```

若您刪除此服務連結角色，之後需要再次建立，您可以在帳戶中使用相同程序重新建立角色。當您建立新的EMR無伺服器應用程式時，EMR無伺服器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無伺服器使用案例的服務連結角色。在 AWS CLI 或 AWS API，建立具有ops.emr-serverless.amazonaws.com服務名稱的服務連結角色。如需詳細資訊，請參閱IAM使用指南中的建立服務連結角色。如果您刪除此服務連結角色，您可以使用此相同的程序以再次建立該角色。

編輯無伺服器EMR的服務連結角色

EMR無伺服器不允許您編輯 AWSServiceRoleForAmazonEMRServerless 服務連結角色，因為各個實體可能會參照該角色。您無法編輯 AWS EMR無伺服器服務連結角色所使用的擁有IAM原則，因為它包含無伺服器需求的所有必要權限EMR。但是，您可以使用編輯角色的描述IAM。

若要使用編輯 AWSServiceRoleForAmazonEMRServerless 服務連結角色的說明 IAM

將下列陳述式新增至需要編輯服務連結角色描述之IAM實體的權限原則。

```
{
    "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使用指南中的編輯服務連結角色。

刪除無伺服器EMR的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。這樣您就不會擁有未被主動監視或維護的未使用實體。不過，您必須先刪除所有區域中的所有EMR無伺服器應用程式，才能刪除服務連結角色。

> **Note**
>
> 當您嘗試刪除與該角色相關聯的資源時，如果EMR無伺服器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。

若要使用刪除 AWSServiceRoleForAmazonEMRServerless 服務連結角色 IAM

將下列陳述式新增至需要刪除服務連結角色之IAM實體的權限原則。

```
{
    "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"}}
}
```

若要使用手動刪除服務連結角色 IAM

使用IAM控制台 AWS CLI，或 AWS API以刪除 AWSServiceRoleForAmazonEMRServerless 服務連結角色。如需詳細資訊，請參閱IAM使用指南中的刪除服務連結角色。

EMR無伺服器服務連結角色的支援區域

EMR無伺服器支援在所有提供服務的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

您可以指定在代表您呼叫其他 服務時，EMR執行 Serverless 任務所能擔任IAM的角色許可。這包括存取 Amazon S3 以取得任何資料來源、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。

範例執行期政策

您可以將執行期政策，例如下列項目，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範例讀取對 Amazon S3 儲存貯體的存取權。
- 完整存取 S3 儲存貯體。
- 建立和讀取 AWS Glue Data Catalog 的存取權。

若要新增對其他 AWS 資源的存取權，例如 DynamoDB ，您需要在建立執行期角色時，在政策中包含這些資源的許可。

```
{
    "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無EMR伺服器任務。若要進一步了解設定許可，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"
                }
            }
}
```

EMR Serverless 的使用者存取政策範例

您可以為使用者設定精細的政策，具體取決於您希望每個使用者在與 EMR Serverless 應用程式互動時執行的動作。下列政策範例可能有助於為使用者設定正確的許可。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 ） 的詳細資訊，請參閱 IAM 使用者指南中的管理IAM政策。

Power 使用者政策

若要授予 EMR Serverless 所有必要的動作，請建立`AmazonEMRServerlessFullAccess`政策並將其連接至必要的IAM使用者、角色或群組。

以下是允許進階使用者建立和修改 EMR Serverless 應用程式，以及執行其他動作的範例政策，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。

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

當您啟用與 的網路連線時VPC，EMRServerless 應用程式會建立 Amazon EC2 彈性網路介面（ENIs）以與 VPC 資源通訊。下列政策可確保僅在無EMR伺服器應用程式的內容中EC2ENIs建立新的 。

> **Note**
>
> 我們強烈建議設定此政策，以確保除了啟動 EMR Serverless 應用程式的情況EC2ENIs之外，使用者無法建立 。

```
{
    "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"
                }
            }
        }
}
```

如果您想要限制對特定子網路的無EMR伺服器存取，可以使用標籤條件來標記每個子網路。此IAM政策可確保無EMR伺服器應用程式只能在允許的子網路EC2ENIs內建立。

```
{
    "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。如需進一步了解，請參閱 針EMR對無伺服器使用服務連結角色。

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無EMR伺服器應用程式上唯讀許可，以及提交和偵錯任務功能的範例政策。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。

```
{
    "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，您可以限制來自一個團隊的使用者，以便他們只能將任務提交至標示其團隊名稱的無EMR伺服器應用程式。

```
{
    "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": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Team": "team-name"
                }
            }
        }
    ]
}
```

標籤型存取控制的政策

您可以使用身分型政策中的條件，根據標籤控制應用程式和任務執行的存取。

下列範例示範使用 條件運算子搭配 EMR Serverless 條件索引鍵的不同案例和方法。這些IAM政策陳述式僅供示範用途，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。如需有關可拒絕標記動作的政策範例，請參閱 拒絕新增和移除標籤的存取權。

下列範例示範以身分為基礎的許可政策，用於控制 EMR Serverless 應用程式允許的 動作。

僅在具有特定標籤值的資源上允許動作

在下列政策範例中，`StringEquals`條件運算子會嘗試`dev`與標籤部門的值相符。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這些動作，則使用者只能使用具有此值之此標籤的應用程式。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:GetApplication"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "emr-serverless:ResourceTag/department": "dev"
        }
      }
    }
  ]
}
```

您也可以使用條件運算子來指定多個標籤值。例如，若要允許department標籤包含 值dev或 之應用程式的動作test，您可以將先前範例中的條件區塊取代為下列。

```
"Condition": {
        "StringEquals": {
          "emr-serverless: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。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "emr-serverless:RequestTag/department": "dev"
        }
```

```
      }
    }
  ]
}
```

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。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication"
      ],
      "Resource": "*",
      "Condition": {
        "Null": {
          "emr-serverless:RequestTag/department": "false"
        }
      }
    }
  ]
}
```

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具有值不是 之標籤的 EMR Serverless 應用程式上的department標籤dev。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "emr-serverless:TagResource",
        "emr-serverless:UntagResource"
      ],
      "Resource": "*",
      "Condition": {
```

```
        "StringNotEquals": {
          "emr-serverless:ResourceTag/department": "dev"
        }
      }
    }
  ]
}
```

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。他們也無法使用 AWS Management Console、 AWS Command Line Interface （ AWS CLI ） 或 來執行任務 AWS API。若要授予使用者對所需資源執行動作的許可，IAM管理員可以建立IAM政策。然後，管理員可以將IAM政策新增至角色，使用者可以擔任角色。

若要了解如何使用這些範例政策文件來建立身分IAM型JSON政策，請參閱 IAM 使用者指南 中的建立IAM政策。

如需 Amazon EMR Serverless 定義之動作和資源類型的詳細資訊，包括ARNs每種資源類型的 格式，請參閱服務授權參考 中的 Amazon EMR Serverless 的動作、資源和條件金鑰。

主題

- 政策最佳實務
- 允許使用者檢視他們自己的許可

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。

身分型政策會判斷是否有人可以在您的帳戶中建立、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 受管政策並邁向最低權限許可 – 若要開始將許可授予您的使用者和工作負載，請使用 AWS 受管政策，將許可授予許多常見使用案例。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。如需詳細資訊，請參閱 IAM 使用者指南 中的 AWS 受管政策或 AWS 受管政策。

- 套用最低權限許可 – 當您使用IAM政策設定許可時， 只會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 套用許可的詳細資訊，請參閱 IAM 使用者指南 中的政策和許可IAM。
- 使用IAM政策中的條件來進一步限制存取：您可以將條件新增至政策，以限制對動作和資源的存取。例如，您可以撰寫政策條件來指定所有請求都必須使用 傳送SSL。如果透過特定 使用服務動作，例如 AWS 服務，您也可以使用 條件來授予其存取權 AWS CloudFormation。如需詳細資訊，請參閱 IAM 使用者指南 中的IAMJSON政策元素：條件。
- 使用 IAM Access Analyzer 驗證您的IAM政策，以確保安全且功能許可 – IAM Access Analyzer 會驗證新的和現有的政策，使政策符合IAM政策語言 （JSON） 和IAM最佳實務。IAM Access Analyzer 提供超過 100 個政策檢查和可操作的建議，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IAM存取分析器政策驗證。
- 需要多重要素身分驗證 （MFA） – 如果您有需要IAM使用者或 根使用者的案例 AWS 帳戶，請開啟 MFA 以獲得額外的安全性。若要在呼叫API操作MFA時要求 ，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 中的設定 MFA受保護的API存取。

如需 中最佳實務的詳細資訊IAM，請參閱 IAM 使用者指南 中的安全最佳實務IAM。

允許使用者檢視他們自己的許可

此範例示範如何建立政策，允許使用者檢視連接至其IAM使用者身分的內嵌和受管政策。此政策包含在主控台上完成此動作或使用 或 AWS CLI 以程式設計方式完成此動作的許可 AWS API。

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

```
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

受 AWS 管政策的 Amazon EMR Serverless 更新

檢視自此服務開始追蹤這些變更以來，Amazon EMR Serverless 受 AWS 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 Document 歷史記錄頁面上的RSS摘要。

變更	描述	日期
A mazonEMRServerless ServiceRolePolicy – 現有政策的更新	Amazon EMR Serverless 已將新的 `SidCloudWatchPolicyStatement` 和 `EC2PolicyStatement` 新增至 A mazonEMRServerless ServiceRolePolicy 政策 。	2024 年 1 月 25 日
A mazonEMRServerless ServiceRolePolicy – 現有政策的更新	Amazon EMR Serverless 新增了許可，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

變更	描述	日期
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

使用下列資訊來協助您診斷和修正使用 Amazon EMR Serverless 和 時可能遇到的常見問題IAM。

主題

- 我無權在 Amazon EMR Serverless 中執行動作
- 我無權執行 iam：PassRole
- 我想要允許 AWS 帳戶外的人員存取我的 Amazon EMR Serverless 資源

我無權在 Amazon EMR Serverless 中執行動作

如果 AWS Management Console 告訴您未獲授權執行動作，則必須聯絡管理員尋求協助。您的管理員是提供您使用者名稱和密碼的人員。

下列範例錯誤會在 `mateojackson` 使用者嘗試使用主控台檢視一個虛構 *`my-example-widget`* 資源的詳細資訊，但卻無虛構 `emr-serverless:`*`GetWidget`* 許可時發生。

```
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。

我無權執行 iam：PassRole

如果您收到錯誤，表示您無權執行`iam:PassRole`動作，則必須更新您的政策，才能將角色傳遞給 Amazon EMR Serverless。

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

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

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

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

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我想要允許 AWS 帳戶外的人員存取我的 Amazon EMR 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。對於支援資源型政策或存取控制清單（ACLs）的服務，您可以使用這些政策來授予人員對資源的存取權。

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 中的身分和存取管理（IAM）。
- 若要了解如何 AWS 帳戶 在您擁有的 資源之間提供存取權，請參閱 IAM 使用者指南 中的在您 AWS 帳戶 擁有的另一個資源中為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 使用者指南 中的提供存取權給第三方 AWS 帳戶 擁有。 IAM
- 若要了解如何透過身分聯合提供存取權，請參閱 IAM 使用者指南 中的為外部驗證的使用者提供存取權（身分聯合）。
- 若要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 中的跨帳戶資源存取IAM。

將 EMR Serverless 與 搭配使用 AWS Lake Formation ，進行精細存取控制

概觀

透過 Amazon 7.2.0 版及更新EMR版本，您可以利用 AWS Lake Formation 將精細存取控制套用至 S3 支援的 Data Catalog 資料表。此功能可讓您設定 的資料表、資料列、資料欄和儲存格層級存取控制

read Amazon EMR Serverless Spark 任務中的查詢。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 Lake Formation。

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

將 EMR Serverless 與 Lake Formation 搭配使用，可讓您針對每個 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 資源設定檔來建立兩個設定檔，以有效執行任務。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 和 考量和限制。

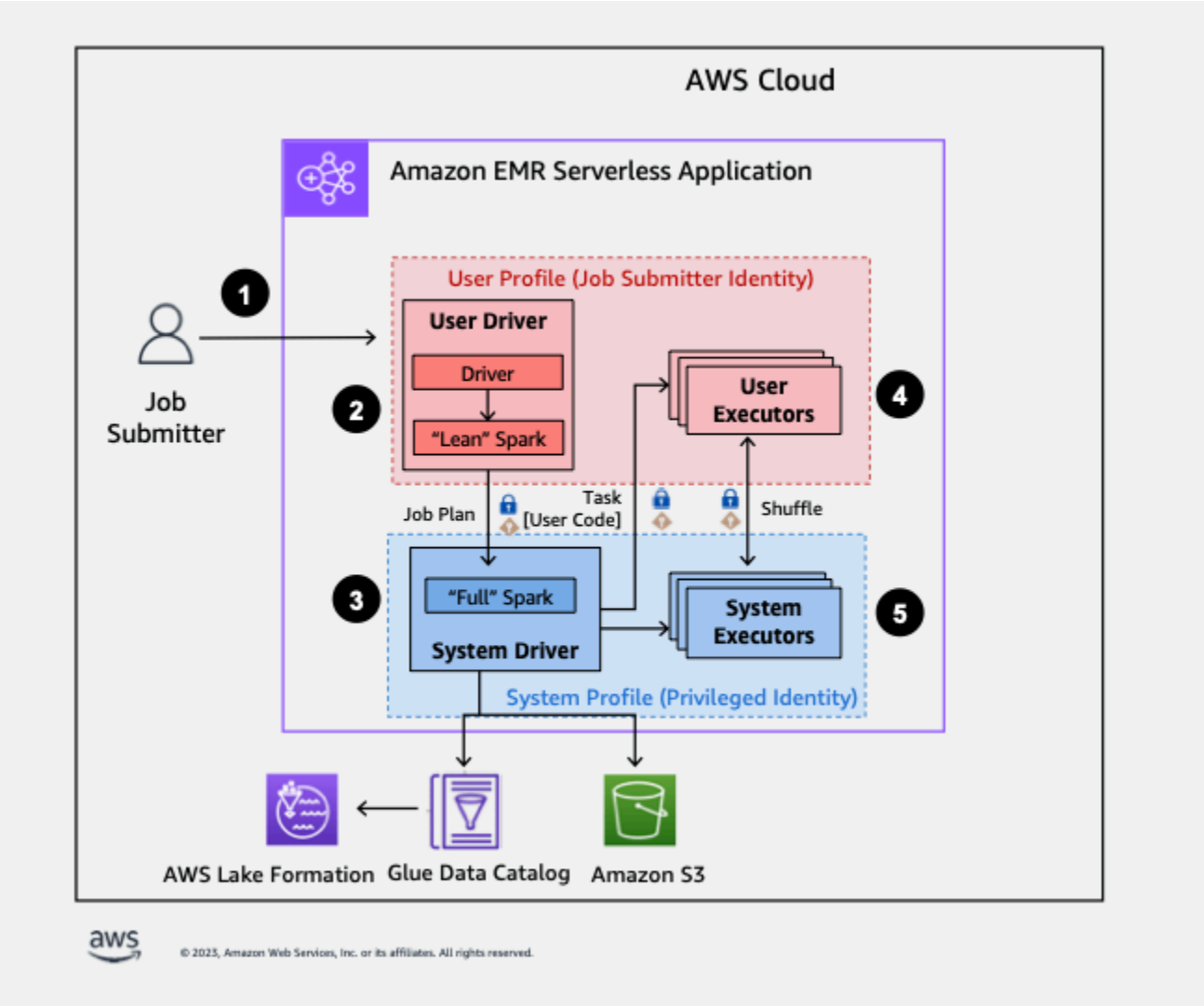
當您搭配 Lake Formation 使用預先初始化的容量時，我們建議您至少有兩個 Spark 驅動程式。每個啟用 Lake Formation 的任務都會使用兩個 Spark 驅動程式，一個用於使用者設定檔，另一個用於系統設定檔。為了獲得最佳效能，相較於不使用 Lake Formation，您應該使用兩倍的已啟用 Lake Formation 任務的驅動程式數目。

當您在 EMR Serverless 上執行 Spark 任務時，也必須考量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`上限組態會套用至使用者和系統執行器。如果您將該數字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卡住，這使得其他執行器在您執行任務時無法運作。

因此，您不會耗盡資源，EMRServerless 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，您也可以設定下列屬性，以最佳化資源配置和整體效能：

- `spark.dynamicAllocation.cachedExecutorIdleTimeout`
- `spark.dynamicAllocation.shuffleTracking.timeout`
- `spark.cleaner.periodicGC.interval`

以下是如何無EMR伺服器存取 Lake Formation 安全政策保護之資料的高階概觀。



1. 使用者將 Spark 任務提交至 AWS Lake Formation已啟用 的 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。使用者驅動程式執行精簡版 Spark，無法啟動任務、請求執行程式、存取 S3 或 Glue Catalog。它建立了一個任務計畫。
3. EMR Serverless 會設定第二個名為系統驅動程式的驅動程式，並在系統設定檔 （具有特殊權限身分） 中執行。EMR 無伺服器會在兩個驅動程式之間設定加密TLS的頻道以進行通訊。使用者驅動程式使用 頻道將任務計劃傳送至系統驅動程式。系統驅動程式不會執行使用者提交的程式碼。它執行完整的 Spark，並與 S3 和資料目錄通訊以進行資料存取。它請求執行器並將任務計畫編譯為一系列執行階段。
4. EMR 然後，無伺服器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都只會在使用者設定檔執行器上執行。

5. 從受 AWS Lake Formation 或套用安全篩選條件的 Data Catalog 資料表讀取資料的階段，會委派給系統執行者。

在 Amazon 中啟用 Lake Formation EMR

若要啟用 Lake Formation，您必須在建立 Serverless 應用程式 時，將 Runtime-configuration 參數的`spark-defaults`分類`spark.emr-serverless.lakeformation.enabled`設為 `true`。EMR

```
aws emr-serverless create-application \
    --release-label emr-7.2.0 \
    --runtime-configuration '{
     "classification": "spark-defaults",
     "properties": {
      "spark.emr-serverless.lakeformation.enabled": "true"
      }
    }' \
    --type "SPARK"
```

您也可以在 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 下取得。

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`設定為 `true`。`spark-submit`

```
--conf spark.emr-serverless.lakeformation.enabled=true
```

任務執行期角色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 許可控制對 Lake Formation、 AWS Glue APIs和資源的存取。雖然您可能擁有 Lake Formation 許可，可存取 Data Catalog （SELECT） 中的資料表，但如果您沒有操作的IAM許可，您的`glue:Get*`API操作會失敗。

以下是如何提供存取 S3 中指令碼的IAM許可、將日誌上傳到 S3、 AWS Glue API許可，以及存取 Lake Formation 的許可的範例政策。

```
{
"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。然後在所需資料表上建立任務執行期角色的許可。如需 Lake Formation 的詳細資訊，請參閱 開發人員指南 中的什麼是 AWS Lake Formation？。AWS Lake Formation

設定 Lake Formation 許可後，您可以在 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

設定 Lake Formation 授予之後，您可以在 EMR Serverless 上提交 Spark 任務。若要執行 Iceberg 任務，您必須提供下列`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
```

開放資料表格式支援

Amazon 7.2.0 EMR版包含支援以 Lake Formation 為基礎的精細存取控制。EMR Serverless 支援 Hive 和 Iceberg 資料表類型。下表說明所有支援的操作。

作業	Hive	Iceberg
DDL 命令	僅具有IAM角色許可	僅具有IAM角色許可
增量查詢	不適用	完全支援
時間歷程查詢	不適用於此資料表格式	完全支援

作業	Hive	Iceberg
中繼資料表	不適用於此資料表格式	受支援，但某些資料表會隱藏。如需詳細資訊，請參閱考量事項和限制。
DML INSERT	僅具有 IAM 許可	僅具有 IAM 許可
DML UPDATE	不適用於此資料表格式	僅具有 IAM 許可
DML DELETE	不適用於此資料表格式	僅具有 IAM 許可
讀取操作	完全支援	完全支援
預存程序	不適用	支援 ，但有 register_table 和 的例外狀況migrate。如需詳細資訊，請參閱考量事項和限制。

考量與限制

當您將 Lake Formation 與 EMR Serverless 搭配使用時，請考慮下列考量和限制。

> **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，則從預先初始化的容量佈建的驅動程式，以及系統驅動程式的數目等於您指定的使用者驅動程式數目。如果您選擇隨需容量，除使用者驅動程式外，EMRServerless 還會啟動系統驅動程式。若要估算與 Lake Formation EMR 的 Serverless 任務相關聯的成本，請使用 AWS Pricing Calculator。

除 AWS GovCloud （美國東部） 和 AWS GovCloud （美國西部） 外，Amazon EMR Serverless with Lake Formation 可在所有支援的 EMR Serverless 區域中使用。

- Amazon EMR Serverless 僅支援 Apache Hive 和 Apache Iceberg 資料表透過 Lake Formation 進行精細存取控制。Apache Hive 格式包括 Parquet、 ORC和 xSV。
- 啟用 Lake Formation 的應用程式不支援使用自訂的無EMR伺服器映像 。

- 您無法關閉 Lake Formation `DynamicResourceAllocation` 任務。
- 您只能將 Lake Formation 與 Spark 任務搭配使用。
- EMR 無伺服器與 Lake Formation 僅支援整個任務中的單一 Spark 工作階段。
- EMR 無伺服器與 Lake Formation 僅支援透過資源連結共用的跨帳戶資料表查詢。
- 不支援下列項目：
  - 彈性分散式資料集（RDD）
  - Spark 串流
  - 寫入 Lake Formation 授予的許可
  - 巢狀資料欄的存取控制
- EMR 無伺服器會封鎖可能會破壞系統驅動程式完整隔離的功能，包括下列項目：
  - UDTs、H iveUDFs和涉及自訂類別的任何使用者定義函數
  - 自訂資料來源
  - 為 Spark 擴充功能、連接器或中繼存放區提供額外的 jar
  - `ANALYZE TABLE` 命令
- 為了強制執行存取控制`EXPLAIN PLAN`和DDL操作，例如`DESCRIBE TABLE`不要公開受限制的資訊。
- EMR Serverless 限制對已啟用 Lake Formation 之應用程式的系統驅動程式 Spark 日誌的存取。由於系統驅動程式執行時具有更多存取權，因此系統驅動程式產生的事件和日誌可能包含敏感資訊。為了防止未經授權的使用者或程式碼存取此敏感資料，無EMR伺服器停用對系統驅動程式日誌的存取。如需疑難排解，請聯絡 AWS 支援。
- 如果您已向 Lake Formation 註冊資料表位置，資料存取路徑會經過 Lake Formation 儲存的憑證，而不論無EMR伺服器任務執行期角色的IAM許可為何。如果您錯誤設定向資料表位置註冊的角色，則使用具有資料表位置 S3 IAM許可的角色提交的任務將會失敗。
- 寫入 Lake Formation 資料表會使用IAM許可，而不是 Lake Formation 授予的許可。如果您的任務執行期角色具有必要的 S3 許可，您可以使用它來執行寫入操作。

以下是使用 Apache Iceberg 時的考量和限制：

- 您只能搭配工作階段目錄使用 Apache Iceberg，不能任意命名目錄。
- 在 Lake Formation 中註冊的 Iceberg 資料表僅支援中繼資料資料表 `history`、`metadata_log_entries`、`snapshots`、`files`、 `manifests`和 `refs`。Amazon 會EMR隱藏可能具有敏感資料的欄，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 儲存的程序。任何資料表都不支援 `register_table`和 `migrate` 程序。
- 建議您使用 Iceberg DataFrameWriterV2 而非 V1。

故障診斷

如需疑難排解解決方案，請參閱下列各節。

日誌

EMR Serverless 使用 Spark 資源設定檔來分割任務執行。EMR Serverless 使用使用者設定檔來執行您提供的程式碼，而系統設定檔則會強制執行 Lake Formation 政策。您可以存取以使用者設定檔執行的任務日誌。

Live UI 和 Spark 歷史記錄伺服器

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已編輯事件。

您可以在 Executors 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，某些資訊會從 Live UI 中編輯，例如輸出記錄的數量。

Lake Formation 許可不足時任務失敗

請確定您的任務執行期角色具有在您存取的資料表DESCRIBE上執行 SELECT和 的許可。

RDD 執行任務失敗

EMR 無伺服器目前不支援啟用 Lake Formation 任務的彈性分散式資料集 （RDD） 操作。

無法存取 Amazon S3 中的資料檔案

請確定您已在 Lake Formation 中註冊資料湖的位置。

安全驗證例外狀況

EMR 無伺服器偵測到安全驗證錯誤。如需協助，請聯絡 AWS 支援。

跨帳戶共用 AWS Glue Data Catalog 和資料表

您可以跨帳戶共用資料庫和資料表，但仍使用 Lake Formation。如需詳細資訊，請參閱 Lake Formation 中的跨帳戶資料共用，以及如何使用 共用 AWS Glue Data Catalog 和資料表跨帳戶 AWS Lake Formation？

工作者間加密

使用 Amazon 6.15.0 及更高EMR版本，您可以在 Spark 任務執行中啟用員工之間的相互TLS加密通訊。啟用時，EMR無伺服器會自動為您的工作執行中佈建的每個 Worker 產生並分配唯一的憑證。當這些 Worker 與交換控制郵件通訊或傳輸隨機資料時，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識別。如果 Worker 無法驗證其他憑證，TLS交換會失敗，且EMR無伺服器會中止它們之間的連線。

如果您將 Lake Formation m 與EMR無伺服器搭配使用，則預設會啟用相互TLS加密。

在EMR無伺服器上啟用相互TLS加密

若要在您的 spark 應用程式上啟用相互TLS加密，請在建立EMR無伺服器應用程式時設定`spark.ssl.internode.enabled`為 true。如果您正在使用 AWS 主控台若要建立EMR無伺服器應用程式，請選擇 [使用自訂設定]，然後展開 [應用程式設定]，然後輸入 `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"
```

如果您想要為個別的 spark 工作執行啟用相互TLS加密，請在使用時設定`spark.ssl.internode.enabled`為 true `spark-submit`。

```
--conf spark.ssl.internode.enabled=true
```

使用 EMR Serverless 進行資料保護的 Secrets Manager

AWS Secrets Manager 是一項秘密儲存服務，可用來保護資料庫憑證、API金鑰和其他秘密資訊。然後，您可以在程式碼中將硬式編碼憑證取代為 Secrets Manager 的API呼叫。這有助於確保檢查程式碼

的人員不會洩露機密，因為機密不存在。如需概觀，若要取得概述，請參閱《AWS Secrets Manager 使用指南》。

Secrets Manager 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 ，以便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

當您將資料儲存在 Secrets Manager 中並在無EMR伺服器組態中使用秘密 ID 時，不會以純文字將敏感組態資料傳遞至無EMR伺服器，並將其公開至外部 APIs。如果您指出金鑰值對包含儲存在 Secrets Manager 中秘密的秘密 ID，EMR則 Serverless 會在將組態資料傳送給工作者執行任務時擷取秘密。

若要指示組態的鍵值對包含儲存在 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，無EMR伺服器會呼叫 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，*`SecretName`*將 取代為要擷取的秘密名稱。包含連字號，後面接著 Secrets Manager 新增至秘密 結尾的六個字元ARN。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.ql",
        "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.ql.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://EXAMPLE-LOG-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.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": "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，請在建立時將下列政策陳述式新增至您的秘密。您必須使用客戶管理的KMS金鑰建立秘密，EMRServerless 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金鑰許可。

在下列政策中，*applicationId*使用應用程式的 ID 取代 。

秘密的資源政策

您必須在 中的秘密資源政策中包含下列許可 AWS Secrets Manager ，才能允許 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無EMR伺服器應用程式 ID 作為條件。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],
      "Principal": {
        "Service": [
          "emr-serverless.amazonaws.com"
        ]
      },
      "Resource": [
        "*"
      ],
```

```
      "Condition": {
        "StringEquals": {
          "aws:SourceArn": "arn:aws:emr-serverless:AWS ##:aws_account_id:/
applications/applicationId"
        }
      }
    }
  ]
}
```

使用下列客戶受管 AWS Key Management Service （ AWS KMS ） 金鑰的政策來建立秘密：

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

輪換是指定期更新秘密。您可以設定 AWS Secrets Manager ，依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。EMR 當任務轉換為執行中狀態時，無伺服器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，以便任務能夠擷取更新的值。

> **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。

將 Amazon S3 Access Grants 與 EMR Serverless 搭配使用

EMR Serverless 的 S3 Access Grants 概觀

使用 Amazon 6.15.0 版及更新EMR版本，Amazon S3 Access Grants 提供可擴展的存取控制解決方案，可讓您用來增強從 EMR Serverless 存取 Amazon S3 資料的能力。如果您的 S3 資料有複雜或大型的許可組態，您可以使用 Access Grants 來擴展使用者、角色和應用程式的 S3 資料許可。

使用 S3 Access Grants 來增強對 Amazon S3 資料的存取，超出執行期角色或連接至可存取無EMR伺服器應用程式的身分IAM角色所授予的許可。

如需詳細資訊，請參閱 Amazon 管理指南中的使用 S3 Access Grants 管理存取權EMR，以及 Amazon Simple Storage Service 使用者指南中的使用 S3 Access Grants 管理存取權。 EMR

本節說明如何啟動使用 S3 Access Grants 提供 Amazon S3 中資料的存取權的無EMR伺服器應用程式。如需將 S3 Access Grants 與其他 Amazon EMR 部署搭配使用的步驟，請參閱下列文件：

- 搭配 Amazon 使用 S3 Access Grants EMR
- 在 Amazon EMR上使用 S3 Access Grants EKS

使用 S3 Access Grants 啟動無EMR伺服器應用程式以進行資料管理

您可以在無EMR伺服器上啟用 S3 Access Grants，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。包含執行 Spark 任務和使用 S3 Access Grants 所需的必要IAM許可，`s3:GetDataAccess`以及`s3:GetAccessGrantsInstanceForPrefix`：

```
{
    "Effect": "Allow",
    "Action": [
    "s3:GetDataAccess",
```

```
    "s3:GetAccessGrantsInstanceForPrefix"
    ],
    "Resource": [     //LIST ALL INSTANCE ARNS THAT THE ROLE IS ALLOWED TO QUERY
         "arn:aws_partition:s3:Region:account-id1:access-grants/default",
         "arn:aws_partition:s3:Region:account-id2:access-grants/default"
    ]
}
```

> **Note**
>
> 如果您為任務執行指定角色，而該IAM角色具有直接存取 S3 的額外許可，則即使使用者沒有 S3 Access Grants 的許可，使用者仍可存取該角色允許的資料。

2. 啟動您的 EMR Serverless 應用程式，Amazon EMR版本標籤為 6.15.0 或更新版本，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 管理指南 中的 S3 Access Grants 與 Amazon 的考量EMR。 EMR

使用記錄 Amazon EMR 無伺服器API呼叫 AWS CloudTrail

Amazon EMR 無伺服器已與 AWS CloudTrail，提供使用者、角色或使用者所採取之動作記錄的服務 AWS EMR無伺服器中的服務。 CloudTrail 將EMR無伺服器的所有API呼叫擷取為事件。擷取的呼叫包括來自EMR無伺服器主控台的呼叫，以及對無伺EMR服API器作業的程式碼呼叫。如果您建立追蹤，您可以啟用持續交付 CloudTrail事件到 Amazon S3 儲存貯體，包括EMR無伺服器事件。如果您未設定追蹤，您仍然可以在 [事件歷程記錄] 中檢視 CloudTrail 主控台中最近的事件。使用收集的資訊 CloudTrail，您可以判斷向EMR無伺服器提出的要求、提出要求的 IP 位址、提出要求的人員、提出要求的時間，以及其他詳細資訊。

若要進一步了解 CloudTrail，請參閱 AWS CloudTrail 用戶指南。

EMR無伺服器資訊 CloudTrail

CloudTrail 已在您的 AWS 帳戶 當您創建帳戶時。當活動在EMR無伺服器中發生時，該活動會與其他活動一起記錄在 CloudTrail 事件中 AWS 事件歷史記錄中的服務事件。您可以查看，搜索和下載最近的事件 AWS 帳戶。 如需詳細資訊，請參閱使用 CloudTrail 事件歷程記錄檢視事件。

在您的事件的持續記錄 AWS 帳戶，包括EMR無伺服器的事件，建立追蹤。追蹤可 CloudTrail 將日誌檔交付到 Amazon S3 儲存貯體。根據預設，當您在主控台中建立追蹤時，追蹤會套用至所有 AWS 區域。 追蹤記錄中所有區域的事件 AWS 對日誌檔進行分割，並將其交付到您指定的 Amazon S3 儲存貯體。此外，您可以配置其他 AWS 進一步分析 CloudTrail 日誌中收集的事件數據並採取行動的服務。如需詳細資訊，請參閱下列內容：

- 建立追蹤的概觀
- CloudTrail 支援的服務與整合
- 設定 Amazon SNS 通知 CloudTrail
- 從多個區域接收 CloudTrail 日誌文件並從多個帳戶接收 CloudTrail 日誌文件

所有EMR無伺服器動作均由記錄， CloudTrail 並將其記錄在EMR無伺服器API參考中。例如，呼叫`StartJobRun`和`CancelJobRun`動作會`CreateApplication`在 CloudTrail 記錄檔中產生項目。

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：

- 是否要求是使用根或 AWS Identity and Access Management (IAM) 使用者認證。
- 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。
- 請求是否由另一個人提出 AWS 服務。

如需詳細資訊，請參閱CloudTrail userIdentity 元素。

瞭解EMR無伺服器記錄檔項目

追蹤是一種組態，可讓事件以日誌檔的形式傳遞到您指定的 Amazon S3 儲存貯體。 CloudTrail 記錄檔包含一或多個記錄項目。事件代表來自任何來源的單一請求，包括有關請求的操作，動作的日期和時間，請求參數等信息。 CloudTrail 日誌文件不是公共API調用的有序堆棧跟踪，因此它們不會以任何特定順序顯示。

下列範例顯示示範`CreateApplication`動作的 CloudTrail 記錄項目。

```
{
    "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"
}
```

適用於 Amazon EMR 無伺服器的合規驗證

第三方稽核員將EMR無伺服器的安全性與合規性評估為多個稽核人員的一部分 AWS 合規計劃，包括以下內容：

- 系統與組織控制 (SOC)
- 支付卡產業資料安全標準 (PCIDSS)
- 聯邦風險和授權管理計劃（FedRAMP）溫和
- Health 保險可攜性與責任法案 (HIPAA)

AWS 提供經常更新的清單 AWS 特定合規計劃範圍內的服務 AWS 合規計劃範圍內的服務。

第三方稽核報告可供您使用下載 AWS Artifact。 如需詳細資訊，請參閱下載報告 AWS Artifact 。

如需關於 AWS 合規方案，請參閱 AWS 合規計劃。

使用EMR無伺服器時，您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。如果您對EMR無伺服器的使用受到符合標準HIPAA，例如PCI，或 Fed RAMP 中等標準， AWS 提供資源以幫助：

- 安全性與合規性快速入門指南，討論以安全性和法規遵循為重點的基準環境部署的架構考量和步驟 AWS.
- AWS 客戶法規遵循指南可協助您透過合規的角度瞭解共同的責任模式。這些指南總結了安全性的最佳做法 AWS 服務 並在多個架構 (包括美國國家標準與技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準化組織 ()) 中對應安全控制指引。ISO
- 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 合規性資源是可能適用於您的產業和位置的工作簿和指南集合。
- AWS Security Hub 為您提供內部安全狀態的全面檢視 AWS 並協助您檢查您是否符合安全性產業標準和最佳做法。
- AWS Audit Manager— 這個 AWS 服務 協助您持續稽核 AWS 使用方式可簡化您管理風險以及遵守法規和業界標準的方式。

Amazon EMR 無伺服器的彈性

所以此 AWS 全球基礎設施是圍繞 AWS 區域和可用區域。 AWS 區域提供多個分開且隔離的實際可用區域，並以低延遲、高輸送量和高度備援聯網功能相互連結。透過可用區域，您可以設計與操作的應用程式和資料庫，在可用區域之間自動容錯移轉而不會發生中斷。可用區域的可用性、容錯能力和擴展能力，均較單一或多個資料中心的傳統基礎設施還高。

如需關於 AWS 區域和可用區域，請參閱 AWS 全球基礎設施。

除了 AWS 全球基礎設施，Amazon EMR 無伺服器提供與 Amazon S3 的整合，EMRFS以協助支援您的資料彈性和備份需求。

Amazon EMR 無伺服器中的基礎設施安全

作為託管服務，Amazon EMR 受到保護 AWS 全球網絡安全。如需相關資訊 AWS 安全服務以及如何 AWS 保護基礎架構，請參 AWS 雲端安全。若要設計您的 AWS 使用基礎架構安全性最佳做法的環境，請參閱安全性支柱中的基礎架構 AWS 架構良好的框架。

您使用 AWS 通過網絡訪問 Amazon EMR 的已發布API呼叫。使用者端必須支援下列專案：

- 傳輸層安全性 (TLS)。我們需要 TLS 1.2 並推薦 TLS 1.3。

- 具有完美前向保密（ ）的密碼套件，例如（短暫的迪菲-赫爾曼PFS）或DHE（橢圓曲線短暫迪菲-赫爾曼）。ECDHE現代系統(如 Java 7 和更新版本)大多會支援這些模式。

此外，請求必須使用存取金鑰 ID 和與IAM主體相關聯的秘密存取金鑰來簽署。或者您可以使用 AWS Security Token Service (AWS STS)，以產生用來簽署要求的臨時安全登入資料。

Amazon EMR 無伺服器中的組態和漏洞分析

AWS 處理基本安全性工作，例如客體作業系統 (OS) 和資料庫修補、防火牆組態和嚴重損壞修復。這些程序已由適當的第三方進行檢閱並認證。如需詳細資訊，請參閱以下 資源：

- 適用於 Amazon EMR 無伺服器的合規驗證
- 共同的責任模型
- Amazon Web Services：安全程序概觀 (白皮書)

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的 。除了標準 AWS 端點之外，有些 AWS 服務 還提供所選區域中的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。

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
Asia Pacific (Hong Kong)	ap-east-1	emr-serverless.ap-east-1.amazonaws.com	HTTPS
亞太區域 (雅加達)	ap-southeast-3	emr-serverless.ap-southeast-3.amazonaws.com	HTTPS
亞太區域 (孟買)	ap-south-1	emr-serverless.ap-south-1.amazonaws.com	HTTPS
亞太區域 (大阪)	ap-northeast-3	emr-serverless.ap-northeast-3.amazonaws.com	HTTPS

區域名稱	區域	端點	通訊協定
亞太區域 (首爾)	ap-northeast-2	emr-serverless.ap-northeast-2.amazonaws.com	HTTPS
亞太區域 (新加坡)	ap-southeast-1	emr-serverless.ap-southeast-1.amazonaws.com	HTTPS
亞太區域 (雪梨)	ap-southeast-2	emr-serverless.ap-southeast-2.amazonaws.com	HTTPS
亞太區域 (東京)	ap-northeast-1	emr-serverless.ap-northeast-1.amazonaws.com	HTTPS
加拿大 (中部)	ca-central-1 （僅限下列可用區域： cac1-az1和 cac1-az2 ）	emr-serverless.ca-central-1.amazonaws.com	HTTPS
歐洲 (法蘭克福)	eu-central-1	emr-serverless.eu-central-1.amazonaws.com	HTTPS

區域名稱	區域	端點	通訊協定
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
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
Middle East (Bahrain)	me-south-1	emr-serverless.me-south-1.amazonaws.com	HTTPS

區域名稱	區域	端點	通訊協定
中東 （UAE）	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
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 您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 帳戶的初始較低配額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？	描述
vCPUs 每個帳戶的並行上限	16	是	目前 中 vCPUs 可同時為 帳戶執行的 數目上限 AWS 區域。
每個帳戶的佇列任務上限	2000	是	目前 中帳戶的佇列任務數目上限 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 50 次交易的爆量。
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 50 次交易的爆量。
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 2 次交易的爆量。
CancelJobRun	每秒 1 筆交易。每秒暴增 25 次交易。
GetJobRun	每秒 10 筆交易。每秒 50 次交易的爆量。
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無伺服器的其他考量事項。

- EMR以下提供無伺服器服務 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)
  - Asia Pacific (Hong Kong)
  - 亞太區域 (雅加達)
  - 亞太區域 (孟買)
  - 亞太區域 (大阪)
  - 亞太區域 (首爾)
  - 亞太區域 (新加坡)
  - 亞太區域 (雪梨)
  - 亞太區域 (東京)
  - 加拿大 (中部)
  - 歐洲 (法蘭克福)
  - 歐洲 (愛爾蘭)
  - 歐洲 (倫敦)
  - 歐洲 (米蘭)
  - Europe (Paris)
  - 歐洲 (西班牙)
  - 歐洲 (斯德哥爾摩)
  - Middle East (Bahrain)
  - 中東 (UAE)
  - 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。

- 工作執行的預設逾時時間為 12 小時。您可以使用`startJobRun`API或中的`executionTimeoutMinutes`性質變更此設定 AWS SDK。如果您希望工作執行永不逾時，可以將其設`executionTimeoutMinutes`定為 0。例如，如果您有串流應用程式，您可以設定`executionTimeoutMinutes`為 0 以允許串流工作持續執行。
- 中的`billedResourceUtilization`屬性會`getJobRun`API顯示彙總 v CPU、記憶體和儲存 AWS 已針對工作執行收費。計費資源包括員工最低使用 1 分鐘，以及每個工作人員超過 20 GB 的額外儲存空間。這些資源不包括閒置預先初始化 Worker 的使用情況。
- 如果沒有VPC連接，作業可以訪問一些 AWS 服務 端點在相同 AWS 區域。 這些服務包括 Amazon S3， AWS Glue，Amazon CloudWatch 日誌， AWS KMS, AWS Security Token Service, Amazon DynamoDB和 AWS Secrets Manager。 您可以啟用VPC連接以訪問其他 AWS 服務 通過 AWS PrivateLink，但您不需要這樣做。若要存取外部服務，您可以使用VPC.
- EMR無伺服器不支援HDFS。Worker 上的本機磁碟是暫時儲存體，EMR無伺服器會在工作執行期間使用它來洗牌和處理資料。
- EMR無伺服器不支援現有emr-dynamodb-connector的.

Amazon EMR 無伺服器發行版本

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。每個版本都包含大數據應用程式、元件和功能，這些功能可讓您在執行任務時部署和設定 Amazon EMR 無伺服器。

使用 Amazon EMR 6.6.0 及更高版本，您可以部署無EMR伺服器。舊EMR版 Amazon 無法使用此部署選項。當您提交工作時，您必須指定下列其中一個支援的版本。

主題

EMR Serverless 7.2.0

下表列出了可用的應用程式版本 EMR Serverless 7.2.0.

應用程式	版本
Apache Spark	3.5.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR無伺服器 7.2.0 版本資訊

- EMR無伺服器的 Lake Formation — 您現在可以使用 AWS Lake Formation ，對 S3 支援的資料目錄資料表套用精細的存取控制。此功能可讓您針對EMR無伺服器 Spark 作業中的讀取查詢，設定表格、列、欄和儲存格層級存取控制。如需詳細資訊，請參閱 the section called “Lake Formation for FGAC” 和 the section called “考量事項”。

EMR Serverless 7.1.0

下表列出了可用的應用程式版本 EMR Serverless 7.1.0.

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

下表列出了可用的應用程式版本 EMR Serverless 7.0.0.

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

下表列出了可用的應用程式版本 EMR Serverless 6.15.0.

應用程式	版本
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR無伺服器 6.15.0 版本資訊

- TLS支援 — 使用 Amazon EMR 無伺服器版本 6.15.0 及更高版本，您可以在 Spark 任務執行中啟用員工之間的相互TLS加密通訊。啟用時，EMRServerless 會自動為每個工作者產生唯一的憑證，這些憑證在工作執行中佈建的工作執行中，工作者會在TLS握手期間互相驗證，並建立加密通道以安全地處理資料。如需有關相互TLS加密的詳細資訊，請參閱 Worker 間加密。

EMR Serverless 6.14.0

下表列出了可用的應用程式版本 EMR Serverless 6.14.0.

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

下表列出了可用的應用程式版本 EMR Serverless 6.13.0.

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

下表列出了可用的應用程式版本 EMR Serverless 6.12.0.

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

下表列出了可用的應用程式版本 EMR Serverless 6.11.0.

應用程式	版本
Apache Spark	3.3.2
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR無伺服器 6.11.0 版本資訊

- 存取其他帳戶中的 S3 資源-使用 6.11.0 及更高版本，您可以設定多個IAM角色，以在存取不同的 Amazon S3 儲存貯體時採用 AWS 來自EMR無伺服器的帳戶。

EMR Serverless 6.10.0

下表列出了可用的應用程式版本 EMR Serverless 6.10.0.

應用程式	版本
Apache Spark	3.3.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR無伺服器 6.10.0 版本資訊

- 對於版本 6.10.0 或更高版本的EMR無伺服器應用程式，內容的預設值為。`spark.dynamicAllocation.maxExecutors` `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

下表列出了可用的應用程式版本 EMR Serverless 6.9.0.

應用程式	版本
Apache Spark	3.3.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR無伺服器 6.9.0 版本資訊

- Amazon Redshift 集成阿帕奇星火包含在 Amazon EMR 版本 6.9.0 及更高版本。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 無服務器上使用亞馬遜紅移集成阿帕奇星火 EMR。
- EMR無伺服器版本 6.9.0 新增支援 AWS 引力 2（臂 64）架構。您可以使用`create-application`和的`architecture`參數`update-application`APIs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無伺服器 Spark 和 Hive 應用程式匯出、匯入、查詢和加入 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

- 如果針對 Apache Spark 使用 Amazon Redshift 整合，並且具有 Parquet 格式的精確度為微秒的 time、timetz、timestamp 或 timestamptz，則連接器會將時間值四捨五入為最接近的微秒值。請使用文字卸載格式 `unload_s3_format` 參數作為一種解決方法。

EMR Serverless 6.8.0

下表列出了可用的應用程式版本 EMR Serverless 6.8.0.

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

下表列出了可用的應用程式版本 EMR Serverless 6.7.0.

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	描述
功能	Tez 排程器現在支援 Tez 工作的預佔，而不是容器的預佔

EMR Serverless 6.6.0

下表列出了可用的應用程式版本 EMR Serverless 6.6.0.

應用程式	版本
Apache Spark	3.2.0
Apache Hive	3.1.2
Apache Tez	0.9.2

EMR無伺服器初始版本說明

- EMR無伺服器支援 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無伺服器支援 Hive 組態分類`hive-site`、`tez-site`、`emrfs-site`、和`core-site`。這種分類可以改變蜂巢的`hive-site.xml`文件，TEZ 的`tez-site.xml`文件，Amazon EMR 的EMRFS設置，或 Hadoop 的`core-site.xml`文件，分別值。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

- 下表列出了蜂巢和 Tez 反向移植。

蜂巢和特茲變化

變更	描述
向後移植	TEZ-4430：固定的問題與財產 `tez.task.launch.cmd-opts`
向後移植	HIVE-25971：修正 Tez 工作關閉延遲，因為開啟快取的執行緒集區

文件歷史記錄

下表說明自上次發行EMR無伺服器以來，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
更新為現有策略。	將新的SidCloudWatchPolicyStatement 和添加EC2PolicyStatement 到 A mazonEMRServerless ServiceRolePolicy 策略中。	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	設定當您從EMR無伺服器 (6.11 及更高版本) 存取不同帳戶中的 Amazon S3 儲存貯體時採用的多個IAM角色	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
更新為默認蜂房屬性	更新hive.driver.disk 、、hive.tez.disk.size 和 tez.grouping.min-size Hive 工作屬性的預設值。hive.tez.auto.reducer.parallelism	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
服務連結角色原則的更新	更新AmazonEMRServerlessServiceRolePolicy_SLR角色以在命名空間中發佈帳戶層級使用情況"AWS/Usage" 。	2023 年 4 月 20 日
EMR Serverless 一般可用性 (GA)	這是EMR無伺服器的第一個公開發行版本。	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。