Amazon EMR 无服务器用户指南

Amazon EMR

Amazon EMR: Amazon EMR 无服务器用户指南

Table of Contents

什么是 Amazon EMR 无服务器？ 1
概念 1
发布版本 1
应用程序 1
任务运行 2
工作线程 2
预先初始化的容量 3
EMR工作室 3
入门的先决条件 4
注册获取 AWS 账户 4
创建具有管理访问权限的用户 4
授予权限 5
授予编程式访问权限 7
设置 AWS CLI 8
打开 控制台 9
开始使用 10
权限 10
存储 10
交互工作负载 10
创建作业运行时角色 11
从控制台入门 16
第 1 步：创建 应用程序 16
步骤 2：提交作业运行或交互式工作负载 17
步骤 3：查看应用程序 UI 和日志 20
步骤 4：清除 20
从这里开始 AWS CLI 20
第 1 步：创建 应用程序 20
步骤 2：提交作业运行 21
步骤 3：查看输出 23
步骤 4：清除 24
与应用程序交互 26
应用程序状态 26
使用 EMR Studio 控制台 27
创建应用程序 27

列出应用程序 28
管理应用程序 28
使用 AWS CLI 29
配置应用程序 30
应用程序行为 30
预先初始化的容量 32
默认应用程序配置 35
自定义图像 40
先决条件 31
步骤 1：使用EMR无服务器基础镜像创建自定义镜像 41
步骤 2：在本地验证映像 42
第 3 步：将图片上传到您的 Amazon ECR 存储库 43
步骤 4：使用自定义映像创建或更新应用程序 43
步骤 5：允许 EMR Serverless 访问自定义镜像存储库 45
注意事项和限制 45
配置VPC访问权限 46
创建应用程序 46
配置应用程序 48
子网规划的最佳实践 48
架构选项 50
使用 x86_64 架构 50
使用 arm64 架构（Graviton） 50
使用 Graviton 启动新应用程序 51
将现有应用程序转换为 Graviton 51
注意事项 52
上传数据 53
先决条件 53
开始使用 S3 Express One Zone 54
运行任务 56
任务运行状态 56
使用 EMR Studio 控制台 57
提交作业 57
查看作业运行 59
使用 AWS CLI 59
使用经过洗牌优化的磁盘 61
主要优势 61

开始使用 61
直播职位 65
注意事项和限制 66
开始使用 67
直播连接器 67
日志管理 70
Spark 职位 70
火花参数 70
火花特性 73
火花示例 77
Hive 职位 78
Hive 参数 78
蜂巢属性 80
Hive 示例 90
作业弹性 91
使用重试策略监控作业 94
使用重试策略进行登录 94
元数据仓配置 94
使用 AWS Glue 数据目录作为元数据库 94
使用外部 Hive 元数据仓 99
跨账户 S3 访问权限 104
先决条件 104
使用 S3 存储桶策略 104
使用代入的角色 105
假设角色示例 108
纠正错误 112
错误：超过了最大允许容量的限制。 112
错误：已超过配置的最大容量。请稍后重试。 112
错误：S3 访问被拒绝。请检查作业运行时角色对所需 S3 资源的 S3 访问权限。 112
错误: ModuleNotFoundError: 未命名模块<module>。请参阅用户指南，了解如何在EMR无服务器中使用 python 库。 113
错误：无法担任执行角色<role name>，因为该角色不存在或未设置所需的信任关系。 113
运行交互式工作负载 114
概述 114
先决条件 114
权限 114

配置 115
注意事项 116
通过 Apache Livy 端点运行交互式工作负载 117
先决条件 117
所需的权限 117
开始使用 118
注意事项 125
日记账记录和监控 126
存储日志 126
托管存储 127
Amazon S3 127
Amazon CloudWatch 128
轮换日志 131
加密日志 132
托管存储 132
Amazon S3 存储桶 133
Amazon CloudWatch 133
所需的权限 133
配置 log4j2 137
log4j2 和 Spark 137
监控 141
申请和职位 141
Spark 引擎指标 148
使用情况指标 151
使用自动化 EventBridge 152
EMR无服务器事件 EventBridge示例 153
标记资源 156
什么是标签？ 156
标记资源 156
标签限制 157
使用标签 157
教程 159
使用 Java 17 159
JAVA_HOME 159
spark-defaults 160
使用 Hudi 161

使用 Iceberg 162
使用 Python 库 162
使用原生 Python 功能 163
构建 Python 虚拟环境 163
将 PySpark 作业配置为使用 Python 库 164
使用不同的 Python 版本 165
使用三角洲湖 OSS 166
亚马逊 6.9.0 及更高EMR版本 166
亚马逊 6.8.0 及更低EMR版本 168
通过 Airflow 提交作业 169
使用 Hive 用户定义的函数 171
使用自定义图片 172
使用自定义 Python 版本 173
使用自定义 Java 版本 173
构建数据科学映像 174
使用 Apache Sedona 处理地理空间数据 174
在 Amazon Redshift 上使用 Spark 175
启动 Spark 应用程序 175
对 Amazon Redshift 进行身份验证 176
对 Amazon Redshift 进行读取和写入 179
注意事项 180
连接到 DynamoDB 181
第 1 步：上传到亚马逊 S3 181
步骤 2：创建 Hive 表 182
步骤 3：复制到 DynamoDB 183
步骤 4：从 DynamoDB 进行查询 185
设置跨账户访问 187
注意事项 189
安全性 190
安全最佳实操 191
采用最低权限原则 191
隔离不受信任的应用程序代码 191
基于角色的访问控制 (RBAC) 权限 191
数据保护 191
静态加密 192
传输中加密 194

Identity and Access Management (IAM) 194
受众 195
使用身份进行身份验证 196
使用策略管理访问 198
EMR无服务器的工作原理 IAM 200
使用服务相关角色 205
Amazon EMR Serverless 的 Job 运行时角色 210
用户访问策略 212
基于标签的访问控制策略 216
基于身份的策略 219
策略更新 221
故障排除 222
Lake Formation FGAC 224
概述 224
工作方式 224
启用Lake Formation 226
启用运行时权限 226
设置运行时权限 228
提交作业运行 228
支持的操作 228
注意事项 229
故障排除 231
工作者间加密 232
在EMR无服务器上启用双向TLS加密 232
用于数据保护的 Secrets Manager 232
秘密是如何运作的 233
创建密钥 233
指定机密引用 233
授予访问密钥的权限 236
轮换秘密 237
用于数据访问控制的 S3 访问授权 238
概述 238
启动应用程序 238
注意事项 239
CloudTrail 用于记录 240
EMR中的无服务器信息 CloudTrail 240

了解EMR无服务器日志文件条目 241
合规性验证 242
弹性 243
基础设施安全性 243
配置和漏洞分析 244
端点和限额 245
服务端点 245
服务限额 249
API限制 250
其他考虑因素 45
发行版 253
EMR Serverless 7.2.0 253
EMR Serverless 7.1.0 254
EMR Serverless 7.0.0 254
EMR Serverless 6.15.0 254
EMR Serverless 6.14.0 255
EMR Serverless 6.13.0 255
EMR Serverless 6.12.0 256
EMR Serverless 6.11.0 256
EMR Serverless 6.10.0 256
EMR Serverless 6.9.0 257
EMR Serverless 6.8.0 258
EMR Serverless 6.7.0 258
发动机特定的更改 258
EMR Serverless 6.6.0 259
文档历史记录 261
..... cclxiii

什么是 Amazon EMR 无服务器？

Amazon EMR Serverless 是亚马逊的部署选项EMR，它提供了无服务器运行时环境。这简化了使用最新开源框架（例如 Apache Spark 和 Apache Hive）的分析应用程序的操作。借EMR助 Serverless，您无需配置、优化、保护或操作集群即可使用这些框架运行应用程序。

EMRServerless 可帮助您避免为数据处理任务配置过度或资源不足。EMRServerless 会自动确定应用程序所需的资源，获取这些资源来处理您的作业，并在任务完成时释放资源。对于应用程序需要在几秒钟内得到响应的用例，例如交互式数据分析，您可以在创建应用程序时预先初始化应用程序所需的资源。

借助 EMR Serverless，您将继续获得 Amazon 的好处EMR，例如开源兼容性、并发性以及针对流行框架的优化运行时性能。

EMRServerless 适用于希望使用开源框架轻松操作应用程序的客户。它提供快速启动作业、自动容量管理和简单的成本控制。

概念

在本节中，我们将介绍EMR无服务器用户指南中出现的EMR无服务器术语和概念。

发布版本

Amazon EMR 版本是一组来自大数据生态系统的开源应用程序。每个版本都包含不同的大数据应用程序、组件和功能，您可以选择这些应用程序、组件和功能让 EMR Serverless 部署和配置这些应用程序，以便它们可以运行您的应用程序。创建应用程序时，必须指定其发行版本。选择要在应用程序中使用的 Amazon EMR 发行版本和开源框架版本。要了解有关预发行版本的更多信息，请参阅Amazon EMR 无服务器发布版本。

应用程序

借助 EMR Serverless，您可以创建一个或多个使用开源分析框架的EMR无服务器应用程序。要创建应用程序，必须指定以下属性：

- 您要使用的开源框架版本的 Amazon EMR 发布版本。要确定您的发行版本，请参阅Amazon EMR 无服务器发布版本。
- 你希望你的应用程序使用的特定运行时，例如 Apache Spark 或 Apache Hive。

创建应用程序后，您可以向应用程序提交数据处理任务或交互式请求。

每个EMR无服务器应用程序都运行在安全的 Amazon Virtual Private Cloud (VPC) 上，与其他应用程序完全不同。此外，你可以使用 AWS Identity and Access Management (IAM) 用于定义哪些用户和角色可以访问应用程序的策略。您还可以指定限制以控制和跟踪应用程序产生的使用成本。

当你需要执行以下操作时，可以考虑创建多个应用程序：

- 使用不同的开源框架
- 针对不同的用例使用不同版本的开源框架
- 从一个版本升级到另一个版本时执行 A/B 测试
- 为测试和生产场景维护单独的逻辑环境
- 通过独立的成本控制和使用情况跟踪，为不同的团队提供独立的逻辑环境
- 将不同的 line-of-business 应用程序分开

EMRServerless 是一项区域性服务，可简化工作负载在一个区域的多个可用区域中运行的方式。要详细了解如何在 EMR Serverless 中使用应用程序，请参阅与应用程序交互。

任务运行

作业运行是提交给EMR无服务器应用程序的请求，该应用程序以异步方式执行并跟踪其完成过程。作业的示例包括您提交给 Apache Hive 应用程序的 HiveQL 查询，或者您提交给 Apache PySpark Spark 应用程序的数据处理脚本。提交作业时，必须指定作业用来访问的运行时角色（在中IAM创作） AWS 资源，例如 Amazon S3 对象。您可以向应用程序提交多个作业运行请求，并且每个作业运行都可以使用不同的运行时角色进行访问 AWS 资源的费用。EMR无服务器应用程序在收到作业后立即开始执行任务，并同时运行多个作业请求。要了解有关 EMR Serverless 如何运行作业的更多信息，请参阅运行任务。

工作线程

EMR无服务器应用程序在内部使用工作人员来执行您的工作负载。这些工作程序的默认大小取决于您的应用程序类型和 Amazon EMR 发布版本。在安排作业运行时，可以覆盖这些大小。

当您提交作业时，EMRServerless 会计算应用程序为该作业所需的资源并安排工作人员。EMRServerless 会将您的工作负载分解为任务，下载映像、配置和设置工作人员，并在任务完成后停用它们。EMRServerless 会根据每个工作阶段的工作负载和并行度自动向上或向下扩展工作人员。这种自动扩展功能使您无需估算应用程序运行您的工作负载所需的工作人员数量。

预先初始化的容量

EMRServerless 提供了预初始化的容量功能，可让工作人员在几秒钟内完成初始化并做好响应准备。这种容量可以有效地为应用程序创建大量工作人员。要为每个应用程序配置此功能，请设置应用程序的`initial-capacity`参数。配置预初始化的容量后，作业可以立即启动，这样您就可以实现迭代应用程序和时间敏感型作业。要了解有关预初始化工作程序的更多信息，请参阅。配置应用程序

EMR工作室

EMRStudio 是可用于管理EMR无服务器应用程序的用户控制台。如果您在创建第一个EMR无服务器应用程序时您的帐户中不存在 EMR Studio，我们会自动为您创建一个 Studio。您可以从亚马逊EMR控制台访问 EMR Studio，也可以通过IAM或IAM身份中心开启身份提供商 (IdP) 的联合访问权限。当您执行此操作时，用户无需直接访问亚马逊EMR控制台即可访问 Studio 并管理EMR无服务器应用程序。要详细了解EMR无服务器应用程序如何与 EMR Studio 配合使用，请参阅从 EMR Studio 控制台与您的应用程序进行交互和。从 EMR Studio 控制台运行作业

开始使用EMR无服务器的先决条件

主题

- 注册获取 AWS 账户
- 创建具有管理访问权限的用户
- 授予权限
- 安装和配置 AWS CLI
- 打开 控制台

注册获取 AWS 账户

如果你没有 AWS 账户，请完成以下步骤来创建一个。

报名参加 AWS 账户

1. 打开https://portal.aws.amazon.com/billing/注册。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当你注册时 AWS 账户，一个 AWS 账户根用户已创建。root 用户可以访问所有内容 AWS 服务 以及账户中的资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。您可以随时前往 https://aws.amazon.com/并选择“我的账户”，查看您当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

在你注册之后 AWS 账户，保护你的 AWS 账户根用户，启用 AWS IAM Identity Center，然后创建一个管理用户，这样你就不会使用 root 用户来执行日常任务。

保护你的 AWS 账户根用户

1. 登录 。AWS Management Console以账户所有者的身份选择 Root 用户并输入你的 AWS 账户 电子邮件地址。在下一页上，输入您的密码。

有关使用 root 用户登录的帮助，请参阅中以 root 用户身份登录 AWS 登录 用户指南。

2. 为您的 root 用户开启多重身份验证 (MFA)。

   有关说明，请参阅为您的MFA设备启用虚拟设备 AWS 账户 用户指南中的 root IAM 用户（控制台）。

创建具有管理访问权限的用户

1. 启用 “IAM身份中心”。

   有关说明，请参阅启用 AWS IAM Identity Center中的 AWS IAM Identity Center 用户指南。

2. 在 IAM Identity Center 中，向用户授予管理访问权限。

   有关使用教程 IAM Identity Center 目录 作为您的身份来源，请参阅使用默认设置配置用户访问权限 IAM Identity Center 目录中的 AWS IAM Identity Center 用户指南。

以具有管理访问权限的用户身份登录

- 要使用您的 Ident IAM ity Center 用户登录URL，请使用您在创建 Ident IAM ity Center 用户时发送到您的电子邮件地址的登录信息。

  有关使用IAM身份中心用户登录的帮助，请参阅登录 AWS 访问中的门户 AWS 登录 用户指南。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个遵循应用最低权限权限的最佳实践的权限集。

   有关说明，请参阅中的创建权限集 AWS IAM Identity Center 用户指南。

2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅中的添加群组 AWS IAM Identity Center 用户指南。

授予权限

在生产环境中，我们建议您使用更精细的策略。有关此类政策的示例，请参阅EMR无服务器的用户访问策略示例。要了解有关访问管理的更多信息，请参阅访问管理 AWS 《IAM用户指南》中的资源。

对于需要在沙盒环境中开始使用 EMR Serverless 的用户，请使用类似于以下内容的策略：

```
{
    "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/*"
```

```
        }
    ]
}
```

要提供访问权限，请为您的用户、组或角色添加权限：

- 中的用户和群组 AWS IAM Identity Center:

  创建权限集合。按照中创建权限集中的说明进行操作 AWS IAM Identity Center 用户指南。
- IAM通过身份提供商管理的用户：

  创建适用于身份联合验证的角色。按照《IAM用户指南》中为第三方身份提供商创建角色（联合）中的说明进行操作。
- IAM用户：
  - 创建您的用户可以担任的角色。按照《用户指南》中为IAM用户创建角色中的IAM说明进行操作。
  - （不推荐使用）将策略直接附加到用户或将用户添加到用户组。按照《用户指南》中向用户（控制台）添加权限中的IAM说明进行操作。

授予编程式访问权限

如果用户想要与之交互，则需要编程访问权限 AWS 在外面 AWS Management Console。 授予编程访问权限的方式取决于正在访问的用户类型 AWS.

要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
人力身份 （在IAM身份中心管理的用户）	使用临时证书签署对的编程请求 AWS CLI, AWS SDKs，或 AWS APIs.	按照您希望使用的界面的说明进行操作。 • 对于 AWS CLI，请参阅配置 AWS CLI 要使用 AWS IAM Identity Center中的 AWS Command Line Interface 用户指南。 • 对于 AWS SDKs、工具和 AWS APIs，请参阅《IAM身

哪个用户需要编程式访问权限？	目的	方式
		份中心身份验证》中的 AWS SDKs和《工具参考指南》。
IAM	使用临时证书签署对的编程请求 AWS CLI, AWS SDKs，或 AWS APIs.	按照使用临时证书中的说明进行操作 AWS 《IAM用户指南》中的资源。
IAM	（不推荐使用） 使用长期凭证签署对的编程请求 AWS CLI, AWS SDKs，或 AWS APIs.	按照您希望使用的界面的说明进行操作。 • 对于 AWS CLI，请参阅中的使用IAM用户凭据进行身份验证 AWS Command Line Interface 用户指南。 • 对于 AWS SDKs和工具，请参阅中的使用长期凭证进行身份验证 AWS SDKs和《工具参考指南》。 • 对于 AWS APIs，请参阅《IAM用户指南》中的管理IAM用户访问密钥。

安装和配置 AWS CLI

如果要使用EMR无服务器APIs，则必须安装最新版本的 AWS Command Line Interface (AWS CLI)。你不需要 AWS CLI 要从 EMR Studio 控制台中使用 EMR Serverless，您可以按照中的从控制台开始使用EMR无服务器步骤开始使用 Serverless，而无需使用。CLI

要设置 AWS CLI

1. 要安装最新版本的 AWS CLI 对于 macOS、Linux 或 Windows，请参阅安装或更新最新版本的 AWS CLI.
2. 要配置 AWS CLI 并安全设置您的访问权限 AWS 服务（包括EMR无服务器），请参阅使用进行快速配置。`aws configure`

3. 要验证设置，请在 DataBrew 命令提示符下输入以下命令。

```
aws emr-serverless help
```

AWS CLI 命令使用默认值 AWS 区域 来自您的配置，除非您使用参数或配置文件进行设置。要设置你的 AWS 区域 使用参数，您可以将该`--region`参数添加到每个命令中。

要设置你的 AWS 区域 使用配置文件时，首先在`~/.aws/config`文件或文件中添加命名的配置`%UserProfile%/.aws/config`文件（适用于 Microsoft Windows）。按照命名配置文件中的步骤进行操作 AWS CLI。 接下来，设置你的 AWS 区域 以及其他设置，其命令与以下示例中的命令类似。

```
[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
```

打开 控制台

本节中大多数以控制台为导向的主题都是从 A ma EMR zon 控制台开始的。如果你还没有登录你的 AWS 账户，登录，然后打开亚马逊EMR控制台，继续下一节继续使用亚马逊EMR。

开始使用 Amazon EMR Serverless

本教程可帮助您在部署示例 Spark 或 Hive 工作负载时开始使用EMR无服务器。您将创建、运行和调试自己的应用程序。我们将在本教程的大部分内容中显示默认选项。

在启动EMR无服务器应用程序之前，请完成以下任务。

主题

- 授予使用EMR无服务器的权限
- 为EMR无服务器准备存储
- 创建 EMR Studio 来运行交互式工作负载
- 创建作业运行时角色
- 从控制台开始使用EMR无服务器
- 从这里开始 AWS CLI

授予使用EMR无服务器的权限

要使用 EMR Serverless，你需要一个用户或IAM角色附加了授予EMR无服务器权限的策略。要创建用户并将相应的策略附加到该用户，请按照中的说明进行操作授予权限。

为EMR无服务器准备存储

在本教程中，您将使用 S3 存储桶存储将使用EMR无服务器应用程序运行的示例 Spark 或 Hive 工作负载的输出文件和日志。要创建存储桶，请按照《亚马逊简单存储服务控制台用户指南》中创建存储桶中的说明进行操作。将对的任何进一步引*DOC-EXAMPLE-BUCKET*用替换为新创建的存储桶的名称。

创建 EMR Studio 来运行交互式工作负载

如果要使用 EMR Serverless 通过 EMR Studio 中托管的笔记本执行交互式查询，则需要指定 S3 存储桶和EMR无服务器创建工作区的最低服务角色。有关设置步骤，请参阅《Amazon EMR 管理指南》中的设置 EMR Studio。有关交互式工作负载的更多信息，请参阅通过 EMR Studio 使用EMR无服务器运行交互式工作负载。

创建作业运行时角色

在 EMR Serverless 中运行的 Job 使用运行时角色，该角色为特定用户提供精细权限 AWS 服务 以及运行时的资源。在本教程中，公有 S3 存储桶托管数据和脚本。存储桶*DOC-EXAMPLE-BUCKET*存储输出。

要设置作业运行时角色，请先使用信任策略创建一个运行时角色，以便 EMR Serverless 可以使用新角色。接下来，将所需的 S3 访问策略附加到该角色。以下步骤将指导您完成整个过程。

Console

1. 导航到 IAM 控制台，网址为：https://console.aws.amazon.com/iam/。
2. 在左侧导航窗格中，选择 Roles（角色）。
3. 选择 Create role（创建角色）。
4. 对于角色类型，选择自定义信任策略并粘贴以下信任策略。这允许提交到您的 Amazon EMR 无服务器应用程序的任务访问其他 AWS 服务 代表你。

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

5. 选择 “下一步” 导航至 “添加权限” 页面，然后选择 “创建策略”。
6. 创建策略页面将在新选项卡上打开。将政策粘贴到JSON下方。

> **Important**
>
> 将以下策略*DOC-EXAMPLE-BUCKET*中的存储桶名称替换为中创建的实际存储桶名称为EMR无服务器准备存储。这是 S3 访问的基本策略。有关更多作业运行时角色示例，请参阅Amazon EMR Serverless 的 Job 运行时角色。

```
{
    "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:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-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 “查看政策” 页面上，输入策略的名称，例如EMRServerlessS3AndGlueAccessPolicy。
8. 刷新 “附加权限策略” 页面，然后选择EMRServerlessS3AndGlueAccessPolicy。
9. 在 “名称、查看和创建” 页面中，在 “角色名称” 中，输入角色的名称，例如EMRServerlessS3RuntimeRole。要创建此IAM角色，请选择创建角色。

CLI

1. 创建一个名为的文件emr-serverless-trust-policy.json，其中包含用于该IAM角色的信任策略。该文件应包含以下策略。

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

2. 创建一个名为的IAM角色EMRServerlessS3RuntimeRole。使用您在上一步中创建的信任策略。

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

在输出中记下 ARN。您在提交工作时使用新角色的，后面称为*job-role-arn*。ARN

3. 创建一个名为的文件emr-sample-access-policy.json来定义工作负载的IAM策略。这提供了对存储在公共 S3 存储桶中的脚本和数据的读取权限以及对的读写访问权限。*DOC-EXAMPLE-BUCKET*

> **Important**
>
> 将以下策略*DOC-EXAMPLE-BUCKET*中的存储桶名称替换为在为EMR无服务器准备存储... 中创建的实际存储桶名称 这是一项基本政策 AWS Glue 和 S3 访问权限。有关更多作业运行时角色示例，请参阅Amazon EMR Serverless 的 Job 运行时角色。

```
{
    "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:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-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. 使用您在步骤 3 中创建EMRServerlessS3AndGlueAccessPolicy的策略文件创建名为的策略。IAM请注意输出ARN中的，因为您将在下一步中使用新策略。ARN

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

请注意输出ARN中的新政策。你将在下一步*policy-arn*中将其替换。

5. 将IAM策略附加EMRServerlessS3AndGlueAccessPolicy到作业运行时角色EMRServerlessS3RuntimeRole。

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

从控制台开始使用EMR无服务器

完成步骤

- 步骤 1：创建EMR无服务器应用程序
- 步骤 2：提交作业运行或交互式工作负载
- 步骤 3：查看应用程序 UI 和日志
- 步骤 4：清除

步骤 1：创建EMR无服务器应用程序

使用EMR无服务器创建新应用程序，如下所示。

1. 登录 AWS Management Console 然后在 https://console.aws.amazon.com/em r 上打开亚马逊EMR控制台。
2. 在左侧导航窗格中，选择 EMRServerless 以导航到EMR无服务器登录页面。
3. 要创建或管理EMR无服务器应用程序，需要 EMR Studio 用户界面。
   - 如果你已经有EMR工作室了 AWS 区域 您要在其中创建应用程序，然后选择管理应用程序以导航到您的 EMR Studio，或者选择要使用的工作室。
   - 如果你在EMR工作室里没有工作室 AWS 区域 在要创建应用程序的地方，选择 “开始”，然后选择 “创建” 并启动 Studio。EMRServerless 会为您创建一个 EMR Studio，以便您可以创建和管理应用程序。
4. 在新选项卡中打开的 C reate Studio 用户界面中，输入应用程序的名称、类型和发布版本。如果您只想运行批处理作业，请选择 “仅对批处理作业使用默认设置”。对于交互式工作负载，请选择对交互式工作负载使用默认设置。您还可以使用此选项在支持交互的应用程序上运行批处理作业。如果需要，可以稍后更改这些设置。

   有关更多信息，请参阅创建工作室。
5. 选择 “创建应用程序” 以创建您的第一个应用程序。

继续下一节步骤 2：提交作业运行或交互式工作负载以提交作业运行或交互式工作负载。

步骤 2：提交作业运行或交互式工作负载

Spark job run

在本教程中，我们使用 PySpark 脚本来计算多个文本文件中唯一单词的出现次数。公有的、只读的 S3 存储桶同时存储脚本和数据集。

运行 Spark 作业

1. 使用以下命令将示例脚本上传`wordcount.py`到您的新存储桶。

```
aws s3 cp s3://us-east-1.elasticmapreduce/emr-containers/samples/wordcount/
scripts/wordcount.py s3://DOC-EXAMPLE-BUCKET/scripts/
```

2. 步骤 1：创建EMR无服务器应用程序完成后，您将进入 EMR Studio 中的应用程序详细信息页面。在那里，选择 “提交作业” 选项。
3. 在 “提交作业” 页面上，完成以下操作。
   - 在名称字段中，输入您要称之为作业运行的名称。
   - 在 “运行时角色” 字段中，输入您在中创建的角色的名称创建作业运行时角色。
   - 在脚本位置字段中，以 S3 的`s3://DOC-EXAMPLE-BUCKET/scripts/wordcount.py`身份输入URI。
   - 在 “脚本参数” 字段中输入`["s3://DOC-EXAMPLE-BUCKET/emr-serverless-spark/output"]`。
   - 在 Spark 属性部分，选择以文本形式编辑，然后输入以下配置。

```
--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. 要开始运行作业，请选择提交作业。
5. 在 Job runs 选项卡中，您应该看到新作业的运行状态为 “正在运行”。

Hive job run

在本教程的这一部分中，我们将创建一个表，插入几条记录，然后运行计数聚合查询。要运行 Hive 作业，请先创建一个包含所有要作为单个任务一部分运行的 Hive 查询的文件，将该文件上传到 S3，然后在启动 Hive 作业时指定此 S3 路径。

运行 Hive 作业

1. 创建一个名为的文件hive-query.ql，其中包含您要在 Hive 作业中运行的所有查询。

```
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. 使用以下命令上传hive-query.ql到您的 S3 存储桶。

```
aws s3 cp hive-query.ql s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-query.ql
```

3. 步骤 1：创建EMR无服务器应用程序完成后，您将进入 EMR Studio 中的应用程序详细信息页面。在那里，选择 “提交作业” 选项。
4. 在 “提交作业” 页面上，完成以下操作。
   - 在名称字段中，输入您要称之为作业运行的名称。
   - 在 “运行时角色” 字段中，输入您在中创建的角色的名称创建作业运行时角色。
   - 在脚本位置字段中，以 S3 的s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-query.ql身份输入URI。
   - 在 Hive 属性部分，选择编辑为文本，然后输入以下配置。

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

   - 在 Job 配置部分，选择编辑为 JSON，然后输入以下内容JSON。

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

5. 要开始运行作业，请选择提交作业。
6. 在 Job runs 选项卡中，您应该看到新作业的运行状态为 “正在运行”。

Interactive workload

在 Amazon EMR 6.14.0 及更高版本中，您可以使用托管在 EMR Studio 中的笔记本电脑在 Serverless 中EMR为 Spark 运行交互式工作负载。有关包括权限和先决条件在内的更多信息，请参阅通过 EMR Studio 使用EMR无服务器运行交互式工作负载。

创建应用程序并设置所需权限后，请按照以下步骤使用 EMR Studio 运行交互式笔记本：

1. 在 EMR Studio 中导航到 “工作区” 选项卡。如果您仍需要配置 Amazon S3 存储位置和 EMRStudio 服务角色，请选择屏幕顶部横幅中的配置 Studio 按钮。
2. 要访问笔记本，请选择一个工作区或创建一个新的工作区。使用快速启动在新选项卡中打开您的工作区。
3. 转到新打开的选项卡。从左侧导航栏中选择 “计算” 图标。选择EMR无服务器作为计算类型。
4. 选择您在上一节中创建的支持交互的应用程序。
5. 在 “运行时角色” 字段中，输入您的EMR无服务器应用程序可以在作业运行中IAM扮演的角色的名称。要了解有关运行时角色的更多信息，请参阅 Amazon EMR Serverless 用户指南中的 Job 运行时角色。
6. 选择 “附加”。这最多可能需要一分钟。附加后，页面将刷新。
7. 选择一个内核并启动一个笔记本。您也可以在 EMR Serverless 上浏览示例笔记本并将其复制到您的工作区。要访问示例笔记本，请导航至左侧导航栏中的**{...}**菜单，然后浏览笔记本文件名`serverless`中包含的笔记本。
8. 在笔记本中，您可以访问驱动程序日志链接和指向 Apache Spark UI 的链接，Apache Spark UI 是一个实时界面，可提供监控作业的指标。有关更多信息，请参阅 Amazon EMR 无服务器用户指南中的监控EMR无服务器应用程序和作业。

当您将应用程序附加到 Studio 工作区时，如果应用程序尚未运行，则该应用程序会自动启动触发。您也可以预先启动应用程序，并在将其连接到工作区之前做好准备。

步骤 3：查看应用程序 UI 和日志

要查看应用程序用户界面，请先确定作业运行情况。根据作业类型，该作业运行的第一行选项中提供了 Spark U I 或 Hive Tez UI 的选项。选择相应的选项。

如果您选择了 Spark UI，请选择 “执行器” 选项卡以查看驱动程序和执行者日志。如果您选择了 Hive Tez 用户界面，请选择所有任务选项卡以查看日志。

任务运行状态显示为成功后，您可以在 S3 存储桶中查看任务的输出。

步骤 4：清除

虽然您创建的应用程序应在闲置 15 分钟后自动停止，但我们仍然建议您释放不打算再次使用的资源。

要删除应用程序，请导航至列出应用程序页面。选择您创建的应用程序，然后选择操作 → 停止以停止该应用程序。应用程序处于STOPPED状态后，选择同一个应用程序，然后选择操作 → 删除。

有关运行 Spark 和 Hive 作业的更多示例，请参阅Spark 职位和。Hive 职位

从这里开始 AWS CLI

步骤 1：创建EMR无服务器应用程序

使用`emr-serverless create-application`命令创建您的第一个EMR无服务器应用程序。您需要指定应用程序类型以及与要使用的应用程序版本关联的 Amazon EMR 发布标签。应用程序的名称是可选的。

Spark

要创建 Spark 应用程序，请运行以下命令。

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

Hive

要创建 Hive 应用程序，请运行以下命令。

```
aws emr-serverless create-application \
```

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

请注意输出中返回的应用程序 ID。您将使用该 ID 启动申请，并在提交工作时使用，后面称为*application-id*.

在继续之前步骤 2：向您的EMR无服务器应用程序提交作业运行，请确保您的应用程序已达到CREATED状态get-applicationAPI。

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

EMRServerless 会创建工作人员来容纳您请求的工作。默认情况下，这些容量是按需创建的，但您也可以通过在创建应用程序时设置initialCapacity参数来指定预初始化的容量。您还可以使用maximumCapacity参数限制应用程序可以使用的总最大容量。要了解有关这些选项的更多信息，请参阅配置应用程序。

步骤 2：向您的EMR无服务器应用程序提交作业运行

现在，您的EMR无服务器应用程序已准备好运行作业。

Spark

在此步骤中，我们使用 PySpark 脚本来计算多个文本文件中唯一单词的出现次数。公有的、只读的 S3 存储桶同时存储脚本和数据集。应用程序将输出文件和日志数据从 Spark 运行时发送到您创建的 S3 存储桶中的/output和/logs目录。

运行 Spark 作业

1. 使用以下命令将我们要运行的示例脚本复制到您的新存储桶中。

   ```
   aws s3 cp s3://us-east-1.elasticmapreduce/emr-containers/samples/wordcount/
   scripts/wordcount.py s3://DOC-EXAMPLE-BUCKET/scripts/
   ```

2. 在以下命令中，*application-id*用您的应用程序 ID 替换。*job-role-arn*用ARN您在中创建的运行时角色替换创建作业运行时角色。替补 *job-run-name* 用你想称之为作业运行的名字。将所有*DOC-EXAMPLE-BUCKET*字符串替换为您创建的 Amazon S3 存储桶，然后/output添加到路径中。这将在您的存储桶中创建一个新文件夹，EMRServerless 可以在其中复制应用程序的输出文件。

```
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://DOC-EXAMPLE-BUCKET/scripts/wordcount.py",
          "entryPointArguments": ["s3://DOC-EXAMPLE-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. 请注意输出中返回的任务运行 ID。在以下步骤中*job-run-id*使用此 ID 进行替换。

Hive

在本教程中，我们创建一个表，插入几条记录，然后运行计数聚合查询。要运行 Hive 作业，请先创建一个包含所有要作为单个任务一部分运行的 Hive 查询的文件，将该文件上传到 S3，然后在启动 Hive 作业时指定此 S3 路径。

运行 Hive 作业

1. 创建一个名为的文件`hive-query.ql`，其中包含您要在 Hive 作业中运行的所有查询。

```
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. 使用以下命令上传`hive-query.ql`到您的 S3 存储桶。

```
aws s3 cp hive-query.ql s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-
query.ql
```

3. 在以下命令中，*application-id*用您自己的应用程序 ID 替换。*job-role-arn*用ARN您在中创建的运行时角色替换创建作业运行时角色。将所有*DOC-EXAMPLE-BUCKET*字符串替换为

您创建的 Amazon S3 存储桶，/output并在路径中/logs添加和。这会在您的存储桶中创建新的文件夹，EMRServerless 可以在其中复制应用程序的输出和日志文件。

```
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"
            }
        }],
        "monitoringConfiguration": {
          "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/logs"
           }
        }
    }'
```

4. 请注意输出中返回的任务运行 ID。在以下步骤中*job-run-id*使用此 ID 进行替换。

步骤 3：查看任务运行的输出

作业运行通常需要 3-5 分钟才能完成。

Spark

您可以使用以下命令检查 Spark 作业的状态。

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

将日志目标设置为s3://*DOC-EXAMPLE-BUCKET*/emr-serverless-spark/logs，则可以在下方找到该特定作业的日志s3://*DOC-EXAMPLE-BUCKET*/emr-serverless-spark/logs/applications/*application-id*/jobs/*job-run-id*。

对EMR于 Spark 应用程序，Serverless 每 30 秒将事件日志推送到 S3 日志目标中的sparklogs文件夹。任务完成后，驱动程序和执行器的 Spark 运行时日志会上传到按工作器类型适当命名的文件夹，例如driver或executor。 PySpark 作业的输出将上传到。s3://*DOC-EXAMPLE-BUCKET*/output/

Hive

您可以使用以下命令检查 Hive 作业的状态。

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

将日志目标设置为s3://*DOC-EXAMPLE-BUCKET*/emr-serverless-hive/logs，则可以在下方找到该特定作业的日志s3://*DOC-EXAMPLE-BUCKET*/emr-serverless-hive/logs/applications/*application-id*/jobs/*job-run-id*。

对EMR于 Hive 应用程序，Serverless 会持续将 Hive 驱动程序上传到 S3 日志目标HIVE_DRIVER的文件夹，并将 Tez 任务日志上传到该TEZ_TASK文件夹。任务运行达到SUCCEEDED状态后，Hive 查询的输出将在您在monitoringConfiguration字段中指定的 Amazon S3 位置中可用。configurationOverrides

步骤 4：清除

完成本教程的操作后，可以考虑删除您创建的资源。我们建议您释放不打算再次使用的资源。

删除您的应用程序

要删除应用程序，请使用以下命令。

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

删除您的 S3 日志存储桶

要删除您的 S3 日志记录和输出存储桶，请使用以下命令。*DOC-EXAMPLE-BUCKET*替换为在为EMR无服务器准备存储... 中创建的 S3 存储桶的实际名称

```
aws s3 rm s3://DOC-EXAMPLE-BUCKET --recursive
aws s3api delete-bucket --bucket DOC-EXAMPLE-BUCKET
```

删除您的作业运行时角色

要删除运行时角色，请将该策略与该角色分离。然后，您可以同时删除角色和策略。

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

要删除该角色，请使用以下命令。

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

要删除附加到该角色的策略，请使用以下命令。

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

有关运行 Spark 和 Hive 作业的更多示例，请参阅Spark 职位和。Hive 职位

与应用程序交互

本节介绍如何通过以下方式与 Amazon EMR 无服务器应用程序进行交互 AWS CLI 以及 Spark 和 Hive 引擎的默认值。

主题

- 应用程序状态
- 从 EMR Studio 控制台与您的应用程序进行交互
- 在上与您的应用程序交互 AWS CLI
- 配置应用程序
- 自定义EMR无服务器镜像
- 配置VPC访问权限
- Amazon EMR 无服务器架构选项

应用程序状态

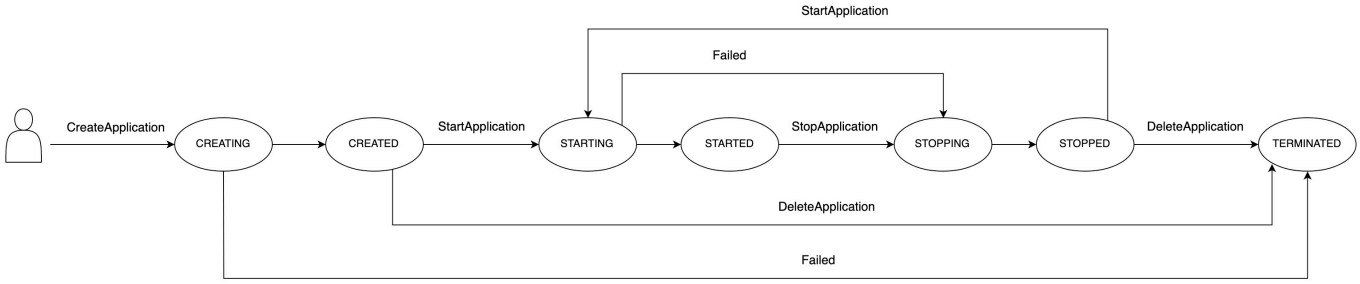
使用 EMR Serverless 创建应用程序时，应用程序运行进入CREATING状态。随后，它将经历以下状态，直至成功（退出并返回代码 0）或失败（退出并返回非零代码）。

应用程序可以具有以下状态：

状态	描述
Creating	该应用程序正在准备中，尚未准备好使用。
Created	应用程序已创建，但尚未配置容量。您可以修改应用程序以更改其初始容量配置。
Starting	应用程序正在启动并正在配置容量。
已开始	该应用程序已准备好接受新工作。应用程序仅在处于此状态时才接受作业。
Stopping	所有作业都已完成，应用程序正在释放其容量。

状态	描述
Stopped	应用程序已停止，并且该应用程序上没有资源在运行。您可以修改应用程序以更改其初始容量配置。
已终止	该应用程序已终止，未出现在您的应用程序列表中。

下图显示了EMR无服务器应用程序状态的轨迹。



从 EMR Studio 控制台与您的应用程序进行交互

在 EMR Studio 控制台中，您可以创建、查看和管理EMR无服务器应用程序。要导航到 EMR Studio 控制台，请按照控制台入门中的说明进行操作。

创建应用程序

在 “创建应用程序” 页面中，您可以按照以下步骤创建EMR无服务器应用程序。

1. 在 “名称” 字段中，输入要调用应用程序的名称。
2. 在 “类型” 字段中，选择 Spark 或 Hive 作为应用程序的类型。
3. 在 “发行版本” 字段中，选择EMR版本号。
4. 在 “架构” 选项中，选择要使用的指令集架构。有关更多信息，请参阅 Amazon EMR 无服务器架构选项。

   - arm64 — 6 4 位ARM架构；使用 Graviton 处理器
   - x86_64 — 64 位 x86 架构；使用基于 x86 的处理器

5. 其余字段有两个应用程序设置选项：默认设置和自定义设置。这些字段是可选的。

   默认设置-默认设置允许您使用预先初始化的容量快速创建应用程序。这包括 Spark 的一个驱动程序和一个执行器，以及 Hive 的一个驱动程序和一个 Tez Task。默认设置不允许与您的网络连接 VPCs。应用程序配置为在闲置 15 分钟后停止，并在提交作业时自动启动。

   自定义设置-自定义设置允许您修改以下属性。

   - 预初始化的容量 — 驾驶员和执行者或 Hive Tez Task 工作人员的数量，以及每个工作人员的规模。
   - 应用程序限制-应用程序的最大容量。
   - 应用程序行为-应用程序的自动启动和自动停止行为。
   - 网络连接-与VPC资源的网络连接。
   - 标签-您可以分配给应用程序的自定义标签。

   有关预初始化容量、应用程序限制和应用程序行为的更多信息，请参阅。配置应用程序有关网络连接的更多信息，请参阅配置VPC访问权限。

6. 要创建应用程序，请选择创建应用程序。

列出应用程序

您可以从 “列出应用程序” 页面查看所有现有的EMR无服务器应用程序。您可以选择应用程序的名称以导航到该应用程序的 “详细信息” 页面。

管理应用程序

您可以从 “列出应用程序” 页面或特定应用程序的 “详细信息” 页面对应用程序执行以下操作。

启动应用程序

选择此选项可手动启动应用程序。

停止应用程序

选择此选项可手动停止应用程序。应用程序应该没有正在运行的作业才能停止。要了解有关应用程序状态转换的更多信息，请参阅应用程序状态。

配置应用程序

从 “配置应用程序” 页面编辑应用程序的可选设置。您可以更改大多数应用程序设置。例如，您可以更改应用程序的发布标签以将其升级到其他版本的 AmazonEMR，也可以将架构从 x86_64 切换到 arm64。其他可选设置与 “创建应用程序” 页面上的 “自定义设置” 部分中的设置相同。有关应用程序设置的更多信息，请参阅创建应用程序。

删除应用程序

选择此选项可手动删除应用程序。必须停止应用程序才能将其删除。要了解有关应用程序状态转换的更多信息，请参阅应用程序状态。

在上与您的应用程序交互 AWS CLI

来自 AWS CLI，您可以创建、描述和删除单个应用程序。您还可以列出所有应用程序，以便一目了然地查看它们。本节介绍如何执行这些操作。有关更多应用程序操作，例如启动、停止和更新应用程序，请参阅 EMRServerless API 参考。有关如何使用EMR无服务器的示例，请API使用 AWS SDK for Java，请参阅我们 GitHub 存储库中的 Java 示例。有关如何使用EMR无服务器的示例，请API使用 AWS SDK for Python (Boto)，请参阅我们 GitHub 仓库中的 Python 示例。

要创建应用程序，请使用`create-application`。必须将`SPARK`或指定`HIVE`为应用程序`type`。此命令返回应用程序的ARN、名称和 ID。

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

要描述应用程序，请使用`get-application`并提供其`application-id`。此命令返回应用程序的状态和容量相关配置。

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

要列出您的所有应用程序，请致电`list-applications`。此命令返回的属性与所有应用程序相同，`get-application`但包括所有应用程序。

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

要删除您的应用程序，请致电delete-application并提供您的application-id。

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

配置应用程序

使用 EMR Serverless，您可以配置所使用的应用程序。例如，您可以设置应用程序可以扩展到的最大容量，配置预初始化的容量以使驱动程序和工作人员做好响应准备，并在应用程序级别指定一组通用的运行时和监控配置。以下页面描述了在使用EMR无服务器时如何配置应用程序。

主题

- 了解应用程序行为
- 预先初始化的容量
- EMR无服务器的默认应用程序配置

了解应用程序行为

应用程序的默认行为

自动启动-默认情况下，应用程序配置为在提交作业时自动启动。您可以关闭此功能。

自动停止 — 默认情况下，应用程序配置为在闲置 15 分钟时自动停止。当应用程序更改为STOPPED状态时，它会释放所有已配置的预初始化容量。您可以修改应用程序自动停止之前的空闲时间，也可以关闭此功能。

最大容量

您可以配置应用程序可以扩展到的最大容量。您可以根据内存 (GB) 和磁盘 (GB) 来指定最大容量。CPU

> **Note**
>
> 我们建议将最大容量配置为与支持的工作人员规模成正比，方法是将工作人员数量乘以其规模。例如，如果您要将应用程序限制为 50 个工作线程，其中 2 vCPUs、16 GB 的内存和 20 GB 的磁盘容量，请将最大容量设置为 100 vCPUs、800 GB 的内存和 1000 GB 的磁盘容量。

支持的工作器配置

下表显示了您可以为 EMR Serverless 指定的支持的工作器配置和大小。您可以根据工作负载的需要为驱动程序和执行程序配置不同的大小。

CPU	内存	默认临时存储
1 v CPU	最小 2 GB，最大 8 GB，以 1 GB 为增量	20 GB-200 GB
2 v CPU	最小 4 GB，最大 16 GB，以 1 GB 为增量	20 GB-200 GB
4 v CPU	最小 8 GB，最大 30 GB，以 1 GB 为增量	20 GB-200 GB
8 v CPU	最小 16 GB，最大 60 GB，以 4 GB 为增量	20 GB-200 GB
16 v CPU	最小 32 GB，最大 120 GB，以 8 GB 为增量	20 GB-200 GB

CPU— 每个工作人员可以有 1、2、4、8 或 16 vCPUs。

内存-每个工作器的内存均在上表中列出的限制范围内，以 GB 为单位指定。Spark 作业有内存开销，这意味着它们使用的内存大于指定的容器大小。此开销由属`spark.driver.memoryOverhead`性和来指定`spark.executor.memoryOverhead`。开销的默认值为容器内存的 10%，最小值为 384 MB。在选择工作人员规模时，应考虑这种开销。

例如，如果您为工作实例选择 4vCPUs ，且预先初始化的存储容量为 30 GB，则应将大约 27 GB 的值设置为 Spark 作业的执行器内存。这样可以最大限度地提高预初始化容量的利用率。可用内存为 27 GB，再加上 27 GB（2.7 GB）的 10%，总共为 29.7 GB。

磁盘 — 您可以为每个工作器配置临时存储磁盘，其大小最小为 20 GB，最大为 200 GB。您只需为每个工作人员配置的超过 20 GB 的额外存储空间付费。

预先初始化的容量

EMRServerless 提供了一项可选功能，可让驱动程序和工作人员进行预初始化并在几秒钟内做好响应准备。这实际上为应用程序创建了一个温暖的工作人员库。此功能称为预初始化容量。要配置此功能，您可以将应用程序的initialCapacity参数设置为要预初始化的工作器数量。使用预先初始化的工作人员容量，作业可以立即启动。当您想要实现迭代应用程序和时间敏感型作业时，这是理想的选择。

提交作业时，如果initialCapacity有来自的工作人员可用，则该作业将使用这些资源开始运行。如果这些工作人员已被其他作业使用，或者该作业需要的资源多于可用资源initialCapacity，则应用程序会请求并获得更多工作人员，但不得超过为应用程序设置的最大资源限制。作业完成运行后，它会释放其使用的工作程序，应用程序的可用资源数量将恢复为initialCapacity。即使在initialCapacity作业完成运行之后，应用程序仍会保留资源。当作业不再需要它们运行initialCapacity时，应用程序会释放多余的资源。

应用程序启动后，预先初始化的容量可用并可供使用。应用程序停止后，预先初始化的容量将变为非活动状态。只有在请求的预初始化容量已创建并可供使用时，应用程序才会进入该STARTED状态。在应用程序处于STARTED状态的整个过程中，EMRServerless 会保持预先初始化的容量可供作业或交互式工作负载使用或使用。该功能可恢复已释放或出现故障的容器的容量。这会保持InitialCapacity参数指定的工作人员数量。没有预初始化容量的应用程序的状态可以立即从CREATED变为。STARTED

如果在一段时间内未使用预先初始化的容量，则可以将应用程序配置为释放预先初始化的容量，默认值为 15 分钟。当您提交新工作时，已停止的应用程序会自动启动。您可以在创建应用程序时设置这些自动启动和停止配置，也可以在应用程序处于CREATED或STOPPED状态时对其进行更改。

您可以为每个 worker 更改InitialCapacity计数并指定计算配置CPU，例如内存和磁盘。由于您无法进行部分修改，因此在更改值时应指定所有计算配置。只有当应用程序处于CREATED或STOPPED状态时，您才能更改配置。

> **Note**
>
> 为了优化应用程序对资源的使用，我们建议将容器大小与预先初始化的容量工作线程大小保持一致。例如，如果您将 Spark 执行器大小配置为 2CPUs，将内存配置为 8 GB，但预先初始化的容量工作器大小为 4CPUs，内存为 16 GB，那么 Spark 执行器在分配给此作业时仅使用一半的工作器资源。

为 Spark 和 Hive 自定义预先初始化的容量

您可以为在特定大数据框架上运行的工作负载进一步自定义预初始化的容量。例如，当工作负载在 Apache Spark 上运行时，您可以指定有多少工作程序以驱动程序的身份启动，有多少工作线程以执行者身份启动。同样，当你使用 Apache Hive 时，你可以指定有多少工作人员开始作为 Hive 驱动程序，以及有多少工作人员应该运行 Tez 任务。

使用预先初始化的容量配置运行 Apache Hive 的应用程序

以下API请求创建了一个运行基于亚马逊EMR版本 emr-6.6.0 的 Apache Hive 的应用程序。该应用程序从 5 个预初始化的 Hive 驱动程序开始，每个驱动程序都有 2 v CPU 和 4 GB 的内存，以及 50 个预初始化的 Tez 任务工作程序，每个驱动程序都有 4 v CPU 和 8 GB 的内存。当 Hive 查询在此应用程序上运行时，它们首先使用预先初始化的工作程序并立即开始执行。如果所有预初始化的工作线程都很忙并且提交了更多 Hive 作业，则应用程序可以扩展到总共有 400 v CPU 和 1024 GB 的内存。您可以选择省略DRIVER或TEZ_TASK工作人员的容量。

```
aws emr-serverless create-application \
  --type "HIVE" \
  --name my-application-name \
  --release-label emr-6.6.0 \
  --initial-capacity '{
    "DRIVER": {
        "workerCount": 5,
        "workerConfiguration": {
            "cpu": "2vCPU",
            "memory": "4GB"
        }
    },
    "TEZ_TASK": {
        "workerCount": 50,
        "workerConfiguration": {
            "cpu": "4vCPU",
            "memory": "8GB"
        }
    }
  }' \
  --maximum-capacity '{
    "cpu": "400vCPU",
    "memory": "1024GB"
  }'
```

使用预先初始化的容量配置运行 Apache Spark 的应用程序

以下API请求创建了一个基于亚马逊 6.6.0 EMR 版本运行 Apache Spark 3.2.0 的应用程序。该应用程序从 5 个预初始化的 Spark 驱动程序开始，每个驱动程序都有 2 v CPU 和 4 GB 的内存，以及 50 个预初始化的执行器，每个执行器都有 4 v CPU 和 8 GB 的内存。当 Spark 作业在此应用程序上运行时，它们首先使用预先初始化的 worker 并立即开始执行。如果所有预初始化的工作程序都处于忙碌状态，并且提交了更多 Spark 作业，则应用程序可以扩展到总共有 400 v CPU 和 1024 GB 的内存。您可以选择省略DRIVER或的容量。EXECUTOR

> **Note**
>
> Spark 将可配置的内存开销（默认值为 10%）添加到驱动程序和执行器请求的内存中。对于要使用预初始化工作线程的作业，初始容量内存配置应大于作业和开销请求的内存。

```
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无服务器的默认应用程序配置

您可以在应用程序级别为在同一应用程序下提交的所有作业指定一组通用的运行时和监控配置。这减少了与需要为每项任务提交相同配置相关的额外开销。

您可以在以下时间点修改配置：

- 在提交作业时声明应用程序级配置。
- 在作业运行期间覆盖默认配置。

以下各节提供了更多详细信息以及进一步背景的示例。

在应用程序级别声明配置

您可以为在应用程序下提交的作业指定应用程序级日志记录和运行时配置属性。

`monitoringConfiguration`

要为通过应用程序提交的作业指定日志配置，请使用字`monitoringConfiguration`段。有关EMR无服务器登录的更多信息，请参阅存储日志。

`runtimeConfiguration`

要指定运行时配置属性（例如）`spark-defaults`，请在`runtimeConfiguration`字段中提供配置对象。这会影响您通过应用程序提交的所有作业的默认配置。有关更多信息，请参阅Hive 配置覆盖参数 和Spark 配置覆盖参数。

可用的配置分类因特定的EMR无服务器版本而异。例如，自定义 Log4j `spark-driver-log4j2` 和的分类仅`spark-executor-log4j2`在 6.8.0 及更高版本中可用。有关特定于应用程序的属性的列表，请参阅Spark 作业属性和。Hive 作业属性

你也可以配置 Apache Log4 j2 属性，AWS Secrets Manager 用于数据保护，以及应用程序级别的 Java 17 运行时。

要在应用程序级别传递 Secrets Manager 密钥，请将以下策略附加到需要使用密钥创建或更新EMR无服务器应用程序的用户和角色。

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

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

有关为机密创建自定义策略的更多信息，请参阅权限策略示例 AWS Secrets Manager中的 AWS Secrets Manager 用户指南。

> **Note**
>
> runtimeConfiguration您在应用程序级别指定的映射到applicationConfiguration中StartJobRunAPI。

声明示例

以下示例说明如何使用声明默认配置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://DOC-EXAMPLE-BUCKET-LOGGING/logs/app-level"
        },
        "managedPersistenceMonitoringConfiguration": {
            "enabled": false
        }
    }'
```

在作业运行期间覆盖配置

您可以使用为应用程序配置和监视配置指定配置替代。StartJobRunAPIEMR然后，Serverless 会合并您在应用程序级别和作业级别指定的配置，以确定任务执行的配置。

合并时的粒度级别如下：

- **ApplicationConfiguration**-例如，分类类型spark-defaults。
- **MonitoringConfiguration**-例如，配置类型s3MonitoringConfiguration。

> **Note**
>
> 您在提供的配置优先级StartJobRun取代您在应用程序级别提供的配置。

有关优先级排名的更多信息，请参阅Hive 配置覆盖参数和Spark 配置覆盖参数。

启动作业时，如果您未指定特定的配置，则该配置将从应用程序继承。如果您在作业级别声明配置，则可以执行以下操作：

- 覆盖现有配置-在StartJobRun请求中提供与您的覆盖值相同的配置参数。
- 添加其他配置-在StartJobRun请求中添加新的配置参数，其中包含您要指定的值。
- 移除现有配置-要删除应用程序运行时配置，请提供要删除的配置的密钥，然后{}为该配置传递一个空声明。我们不建议移除任何包含作业运行所需参数的分类。例如，如果您尝试删除 Hive 作业的必需属性，则该作业将失败。

  要删除应用程序监控配置，请使用适用于相关配置类型的相应方法：

  - **cloudWatchLoggingConfiguration**-要移除cloudWatchLogging，请将启用标志传递为false。
  - **managedPersistenceMonitoringConfiguration**-要移除托管持久性设置并回退到默认的启用状态，请{}为配置传递一个空声明。
  - **s3MonitoringConfiguration**-要删除s3MonitoringConfiguration，请{}为配置传递一个空声明。

示例覆盖

以下示例显示了您在提交作业期间可以执行的不同操作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://DOC-EXAMPLE-BUCKET-OUTPUT/wordcount_output"]
        }
    }' \
    --configuration-overrides '{
        "applicationConfiguration": [
            {
                // Override existing configuration for spark-defaults in the
 application
                "classification": "spark-defaults",
                "properties": {
                    "spark.driver.cores": "2",
```

```
                    "spark.executor.cores": "1",
                    "spark.driver.memory": "4G",
                    "spark.executor.memory": "4G"
                }
            },
            {
                // Add configuration for spark-executor-log4j2
                "classification": "spark-executor-log4j2",
                "properties": {
                    "rootLogger.level": "error",
                    "logger.IdentifierForClass.name": "classpathForSettingLogger",
                    "logger.IdentifierForClass.level": "info"
                }
            },
            {
                // Remove existing configuration for spark-driver-log4j2 from the
application
                "classification": "spark-driver-log4j2",
                "properties": {}
            }
        ],
        "monitoringConfiguration": {
            "managedPersistenceMonitoringConfiguration": {
                // Override existing configuration for managed persistence
                "enabled": true
            },
            "s3MonitoringConfiguration": {
                // Remove configuration of S3 monitoring
            },
            "cloudWatchLoggingConfiguration": {
                // Add configuration for CloudWatch logging
                "enabled": true
            }
        }
    }'
```

在执行任务时，将根据和中描述的优先级覆盖等级应用以下分类Hive 配置覆盖参数和配置Spark 配置覆盖参数。

- 分类spark-defaults将使用在作业级别指定的属性进行更新。此分类仅StartJobRun考虑中包含的属性。
- 该分类spark-executor-log4j2将添加到现有的分类列表中。

- 分类spark-driver-log4j2将被删除。
- 的配置managedPersistenceMonitoringConfiguration将使用作业级别的配置进行更新。
- 的配置s3MonitoringConfiguration将被删除。
- 的配置cloudWatchLoggingConfiguration将添加到现有的监视配置中。

自定义EMR无服务器镜像

从 Amazon EMR 6.9.0 开始，您可以使用自定义映像通过 Amazon Serverless 将应用程序依赖项和运行时环境打包到单个容器中。EMR这简化了管理工作负载依赖关系的方式，并使您的软件包更具可移植性。当您自定义EMR无服务器映像时，它具有以下好处：

- 安装和配置针对您的工作负载进行了优化的软件包。这些软件包可能不会在 Amazon EMR 运行时环境的公开发布中广泛提供。
- 将 EMR Serverless 与组织内当前已建立的构建、测试和部署流程（包括本地开发和测试）集成。
- 应用既定的安全流程，例如图像扫描，以满足组织内的合规性和监管要求。
- 允许您在应用程序中使用自己的JDK和 Python 版本。

EMRServerless 提供的图像可供您在创建自己的图像时用作基础。基础映像提供了映像与 S EMR erverless 交互的必需的 jar、配置和库。您可以在 Amazon ECR 公共图库中找到基础图片。使用与您的应用程序类型（Spark 或 Hive）和发布版本相匹配的图像。例如，如果您在 Amazon EMR 版本 6.9.0 上创建应用程序，请使用以下图片。

类型	图像
Spark	public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest
Hive	public.ecr.aws/emr-serverless/hive/emr-6.9.0:latest

先决条件

在创建EMR无服务器自定义映像之前，请完成以下先决条件。

1. 在同一个ECR仓库中创建一个 Amazon 存储库 AWS 区域 用于启动EMR无服务器应用程序。要创建 Amazon ECR 私有仓库，请参阅创建私有仓库。
2. 要授予用户访问您的 Amazon ECR 存储库的权限，请向使用该存储库中的图像创建或更新EMR无服务器应用程序的用户和角色添加以下策略。

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

有关亚马逊ECR基于身份的策略的更多示例，请参阅 Amazon Elastic Contain er Registry 基于身份的策略示例。

步骤 1：使用EMR无服务器基础镜像创建自定义镜像

首先，创建一个 Dockerfile，该文件以使用您的首选基础映像的FROM指令开头。在FROM说明之后，您可以添加要对图像进行的任何修改。基础图像会自动将设置USER为hadoop。此设置可能不具有您所包含的所有修改的权限。解决方法是，将设置USER为root，修改您的图像，然后将其设置USER回为hadoop:hadoop。要查看常见用例的示例，请参阅在 S EMR erverless 中使用自定义镜像。

```
# 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
```

获得 Dockerfile 后，使用以下命令构建镜像。

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

步骤 2：在本地验证映像

EMRServerless 提供了一个离线工具，可以静态检查您的自定义映像，以验证基本文件、环境变量和正确的映像配置。有关如何安装和运行该工具的信息，请参阅 Amazon EMR Serverless 镜像CLI GitHub。

安装该工具后，运行以下命令来验证映像：

```
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
```

您应该会看到类似于以下内容的输出。

```
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.
----------------------------------------------------------------
```

第 3 步：将图片上传到您的 Amazon ECR 存储库

使用以下命令将您的亚马逊ECR图片推送到您的亚马逊ECR存储库。确保您拥有将图像推送到存储库的正确IAM权限。有关更多信息，请参阅 Amazon ECR 用户指南中的推送图片。

```
# 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
```

步骤 4：使用自定义映像创建或更新应用程序

选择 AWS Management Console 选项卡或 AWS CLI 根据您想要的应用程序启动方式选项卡，然后完成以下步骤。

Console

1. 通过 https://console.aws.amazon.com/em r 登录 EMR Studio 控制台。导航到您的应用程序，或者按照创建应用程序中的说明创建新应用程序。
2. 要在创建或更新EMR无服务器应用程序时指定自定义映像，请在应用程序设置选项中选择自定义设置。
3. 在 “自定义图像设置” 部分中，选中 “在此应用程序中使用自定义图像” 复选框。
4. 将 Amazon ECR 图片URI粘贴到 “图片 URI” 字段中。EMRServerless 将此映像用于应用程序的所有工作器类型。或者，您可以选择不同的自定义图片，然后URIs为每种工作人员类型粘贴不同的 Amazon ECR 图片。

CLI

- 使用`image-configuration`参数创建应用程序。EMRServerless 将此设置应用于所有工作器类型。

```
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"
}'
```

要为每种工作器类型创建具有不同图像设置的应用程序，请使用worker-type-specifications参数。

```
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"
        }
    }
}'
```

要更新应用程序，请使用image-configuration参数。EMRServerless 将此设置应用于所有工作器类型。

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

步骤 5：允许 EMR Serverless 访问自定义镜像存储库

将以下资源策略添加到 Amazon ECR 存储库中，以允许 EMR Serverless 服务主体使用来自此存储库的`getdescribe`、和`download`请求。

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

作为安全最佳实践，请在存储库策略中添加`aws:SourceArn`条件密钥。IAM全局条件密钥`aws:SourceArn`可确保 EMR Serverless 仅将存储库用于应用程序ARN。有关 Amazon ECR 存储库策略的更多信息，请参阅创建私有仓库。

注意事项和限制

使用自定义图像时，请考虑以下几点：

- 使用与您的应用程序的类型（ Spark 或 Hive ）和发布标签（ 例如`emr-6.9.0` ）相匹配的正确基础映像。
- EMR无服务器会忽略 `[CMD]` Docker 文件中的`[ENTRYPOINT]`指令。使用 Docker 文件中的常用指令，例如`[COPY]`[`RUN]`、和。`[WORKDIR]`

- 创建自定义映像TEZ_HOME时 JAVA_HOME SPARK_HOMEHIVE_HOME，不应修改环境变量、、。
- 自定义图像的大小不能超过 5 GB。
- 如果您修改 Amazon EMR 基础映像中的二进制文件或 jar，则可能会导致应用程序或任务启动失败。
- Amazon ECR 存储库应该在同一个仓库中 AWS 区域 用于启动EMR无服务器应用程序。

配置VPC访问权限

您可以将EMR无服务器应用程序配置为连接到您的数据存储，例如 Amazon Redshift 集群VPC、Amazon 数据库或带有终端节点的 RDS Amazon S3 存储桶VPC。您的EMR无服务器应用程序与您的VPC内部数据存储具有出站连接。默认情况下，EMRServerless 会阻止对您的应用程序的入站访问以提高安全性。

> **Note**
>
> 如果要为应用程序使用外部 Hive 元数据仓数据库，则必须配置VPC访问权限。有关如何配置外部 Hive 元数据仓的信息，请参阅元数据仓配置。

创建应用程序

在创建应用程序页面上，您可以选择自定义设置并指定EMR无服务器应用程序可以使用的子网和安全组。VPC

VPCs

选择包含您的数据存储的虚拟私有云 (VPC) 的名称。“创建应用程序” 页面列出了您选择的所有VPCs应用程序 AWS 区域.

子网

在中选择包含您的数据存储VPC的子网。创建应用程序页面列出了您的VPC中数据存储的所有子网。

所选子网必须是私有子网。这意味着子网的关联路由表不应有 Internet 网关。

要实现到 Internet 的出站连接，子网必须有使用网NAT关的出站路由。要配置NAT网关，请参阅使用NAT网关。

对于 Amazon S3 连接，子网必须配置NAT网关或VPC终端节点。要配置 S3 VPC 终端节点，请参阅创建网关终端节点。

用于连接到其他 AWS 服务 在VPC外部（例如 Amazon DynamoDB），您必须配置终端节点VPC或网关。NAT配置VPC终端节点 AWS 服务，请参阅使用VPC端点。

工作人员VPC可以通过出站流量连接到您内部的数据存储。默认情况下，EMRServerless 会阻止工作人员的入站访问以提高安全性。

当你使用时 AWS Config，EMRServerless 会为每位工作人员创建弹性网络接口项目记录。为避免与该资源相关的成本，请考虑关闭 `AWS::EC2::NetworkInterface` AWS Config.

> **Note**
>
> 我们建议您在多个可用区中选择多个子网。这是因为您选择的子网决定了可供EMR无服务器应用程序启动的可用区。每个 worker 都将使用其启动所在子网上的 IP 地址。请确保指定的子网有足够的 IP 地址来满足您计划启动的工作程序的数量。有关子网规划的更多信息，请参阅the section called “子网规划的最佳实践”。

安全组

选择一个或多个可以与您的数据存储进行通信的安全组。创建应用程序页面列出了您的所有安全组VPC。EMRServerless 将这些安全组与连接到您的VPC子网的弹性网络接口相关联。

> **Note**
>
> 我们建议您为EMR无服务器应用程序创建单独的安全组。这使得隔离和管理网络规则更加高效。例如，要与 Amazon Redshift 集群通信，您可以定义 Redshift 和EMR无服务器安全组之间的流量规则，如以下示例所示。

Example 示例 — 与亚马逊 Redshift 集群的通信

1. 为来自其中一个EMR无服务器安全组的 Amazon Redshift 安全组的入站流量添加规则。

类型	协议	端口范围	来源
全部 TCP	TCP	5439	`emr-serverless-security-group`

2. 为来自其中一个EMR无服务器安全组的出站流量添加规则。您可以通过两种方式之一来执行此操作。首先，您可以打开所有端口的出站流量。

类型	协议	端口范围	目标位置
所有流量	TCP	ALL	0.0.0.0/0

或者，您可以限制流向 Amazon Redshift 集群的出站流量。只有当应用程序必须与 Amazon Redshift 集群进行通信而无需进行其他通信时，这才有用。

类型	协议	端口范围	来源
全部 TCP	TCP	5439	`redshift-security-group`

配置应用程序

您可以从配置应用程序页面更改现有EMR无服务器应用程序的网络配置。

查看作业运行详情

在 Job 运行详细信息页面上，您可以查看任务在特定运行中使用的子网。请注意，作业仅在从指定子网中选择的一个子网中运行。

子网规划的最佳实践

AWS 资源是在子网中创建的，子网是 Amazon 中可用 IP 地址的子集VPC。例如，网络掩码为 /16 的 a 最多有 65,536 个可用 IP 地址，可以使用子网掩码将其分成多个较小的网络。VPC例如，您可以将此

范围分成两个子网，每个子网都使用 /17 掩码和 32,768 个可用 IP 地址。子网位于可用区内，不能跨越多个区域。

在设计子网时，应牢记您的EMR无服务器应用程序扩展限制。例如，如果您的应用程序请求 4 个 vCpu 工作线程并且最多可以扩展到 4,000 个vCpu，那么您的应用程序最多需要 1,000 个工作线程才能获得 1,000 个网络接口。我们建议您跨多个可用区创建子网。这样，当可用区出现故障时，EMRServerless 可以重试您的任务或在不同的可用区中配置预先初始化的容量。因此，至少两个可用区中的每个子网应有超过 1,000 个可用 IP 地址。

您需要掩码大小小于或等于 22 的子网才能配置 1,000 个网络接口。任何大于 22 的口罩都不符合要求。例如，子网掩码 /23 提供 512 个 IP 地址，而掩码为 /22 提供 1024 个，掩码为 /21 则提供 2048 个 IP 地址。以下是 4 个子网的示例，这些子网的掩码为 VPC /16，网络掩码为 /16，可以分配给不同的可用区。可用的 IP 地址和可用的 IP 地址之间存在五个差异，因为每个子网中的前四个 IP 地址和最后一个 IP 地址由预留 AWS.

子网 ID	子网地址	子网掩码	IP 地址范围	可用的 IP 地址	可用的 IP 地址
1	10.0.0.0	255.255.252.0/22	10.0.0.0-10.0.3.255	1024	1,019
2	10.0.4.0	255.255.252.0/22	10.0.4.0-10.0.7.255	1024	1,019
3	10.0.8.0	255.255.252.0/22	10.0.4.0-10.0.7.255	1024	1,019
4	10.0.12.0	255.255.252.0/22	10.0.12.0 -10.0.15.255	1024	1,019

您应该评估您的工作负载是否最适合规模较大的员工。使用更大的工作线程需要更少的网络接口。例如，使用应用程序扩展限制为 4,000 的 16 个vCpu 工作线程最多 vCpu 需要 250 个工作线程，总共有 250 个可用 IP 地址来配置网络接口。您需要掩码大小小于或等于 24 的多个可用区域中的子网才能配置 250 个网络接口。任何大于 24 的掩码大小都提供少于 250 个 IP 地址。

如果您在多个应用程序之间共享子网，则每个子网的设计都应考虑到所有应用程序的集体扩展限制。例如，如果您有 3 个应用程序请求 4 个 vCpu 工作人员，并且每个应用程序可以扩展到 4000 个，基于 12,000 个 vCpu vCpu账户级别的服务配额，则每个子网将需要 3000 个可用 IP 地址。如果您要使

用的 IP 地址数量不足，请尝试增加可用 IP 地址的数量。VPC您可以通过将其他无类域间路由 (CIDR) 块与您的关联来实现此目的。VPC有关更多信息，请参阅 Amazon VPC 用户指南VPC中的将其他 IPv4CIDR区块与您的关联。

您可以使用在线提供的众多工具之一来快速生成子网定义并查看其可用的 IP 地址范围。

Amazon EMR 无服务器架构选项

Amazon EMR Serverless 应用程序的指令集架构决定了该应用程序运行任务所使用的处理器类型。亚马逊为您的应用程序EMR提供了两个架构选项：x86_64 和 arm64。EMRServerless 会在最新一代的实例可用时自动更新到这些实例，因此您的应用程序无需您付出额外努力即可使用较新的实例。

主题

- 使用 x86_64 架构
- 使用 arm64 架构（Graviton）
- 启动支持 Graviton 的新应用程序
- 将现有应用程序配置为使用 Graviton
- 使用 Graviton 时的注意事项

使用 x86_64 架构

x86_64 架构也被称为 x86 64 位或 x64。 x86_64 是无服务器应用程序的默认选项。EMR该架构使用基于 x86 的处理器，并且与大多数第三方工具和库兼容。

大多数应用程序都与 x86 硬件平台兼容，并且可以在默认 x86 _64 架构上成功运行。但是，如果您的应用程序与64位兼容ARM，则可以切换到 arm64 以使用Graviton处理器来提高性能、计算能力和内存。在 arm64 架构上运行实例的成本要低于在 x86 架构上运行相同大小的实例的成本。

使用 arm64 架构（Graviton）

AWS Graviton 处理器由以下用户定制设计 AWS 采用 64 位 ARM Neoverse 内核，利用 arm64 架构（也称为 Arch64 或 64 位）。ARM这些区域有： AWS Serv EMR erless 上提供的 Graviton 系列处理器包括 Graviton3 和 Graviton2 处理器。与在 x86_64 架构上运行的同等工作负载相比，这些处理器为 Spark 和 Hive 工作负载提供了卓越的性价比。EMR当最新一代处理器可用时，Serverless 会自动使用最新一代的处理器，您无需付出任何努力即可升级到最新一代的处理器。

启动支持 Graviton 的新应用程序

使用以下方法之一启动使用 arm64 架构的应用程序。

AWS CLI

要使用 Graviton 处理器启动应用程序，请访问 AWS CLI，在中指定ARM64为`architecture`参数`create-application`API。在其他参数中为您的应用程序提供适当的值。

```
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

要使用 EMR Studio 的 Graviton 处理器启动应用程序，请在创建或更新应用程序时选择 arm64 作为架构选项。

将现有应用程序配置为使用 Graviton

您可以将现有的 Amazon EMR 无服务器应用程序配置为使用 Graviton (arm64) 架构，SDK AWS CLI，或 EMR Studio。

将现有应用程序从 x86 转换为 arm64

1. 确认您使用的是最新的主要版本 AWS CLI/SDK，它支持该`architecture`参数。
2. 确认没有作业在运行，然后停止应用程序。

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

3. 要更新应用程序以使用 Graviton，请在中ARM64为`architecture`参数指定。`update-application` API

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

```
 --architecture 'ARM64' \
 --region us-west-2
```

4. 要验证应用程序的CPU架构是否为现在ARM64，请使用`get-application`API。

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

5. 准备就绪后，重新启动应用程序。

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

使用 Graviton 时的注意事项

在启动使用 arm64 支持 Graviton 的EMR无服务器应用程序之前，请确认以下内容。

库兼容性

选择 Graviton (arm64) 作为架构选项时，请确保第三方软件包和库与 64 位架构兼容。ARM有关如何将 Python 库打包到与所选架构兼容的 Python 虚拟环境中的信息，请参阅将 Python 库与EMR无服务器一起使用。

要详细了解如何将 Spark 或 Hive 工作负载配置为使用 64 位ARM，请参阅 AWS Graviton 入门存储库已开启。 GitHub该存储库包含基本资源，可以帮助您开始使用ARM基于Graviton。

使用EMR无服务器将数据导入 S3 Express One 区域

在 Amazon 7.2.0 及更高EMR版本中，您可以将EMR无服务器与 Amazon S3 Express One Zone 存储类配合使用，以提高运行任务和工作负载时的性能。S3 Express One Zone 是一种高性能的单区域 Amazon S3 存储类别，可为大多数对延迟敏感的应用程序提供一致的、个位数毫秒的数据访问权限。S3 Express One Zone 在其发布时，提供了 Amazon S3 中延迟最低、性能最高的云对象存储。

先决条件

- S3 Express One 区域权限 — 当 S3 Express One Zone 最初对 S3 对象执行诸如GETLIST、或PUT之类的操作时，存储类会CreateSession代表您调用。您的IAM策略必须允许该s3express:CreateSession权限，以便 S3A 连接器可以调用CreateSessionAPI。有关使用此权限的示例策略，请参阅 开始使用 S3 Express One Zone。
- S3A 连接器 — 要将 Spark 配置为访问使用 S3 Express One Zone 存储类的 Amazon S3 存储桶中的数据，您必须使用 Apache Hadoop 连接器 S3A。 要使用连接器，请确保所有 S3 都URIs使用该s3a方案。如果他们没有使用，则可以更改用于 s3 和 s3n 方案的文件系统实施。

要更改 s3 方案，请指定以下集群配置：

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

要更改 s3n 方案，请指定以下集群配置：

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

```
  }
]
```

开始使用 S3 Express One Zone

请按照以下步骤开始使用 S3 Express One Zone。

1. 创建VPC终端节点。将终端节点 `com.amazonaws.us-west-2.s3express`添加到终VPC端节点。
2. 按照 Amazon EMR Serverless 入门，创建带有亚马逊EMR版本标签 7.2.0 或更高版本的应用程序。
3. 将您的应用程序配置为使用新创建的VPC终端节点、私有子网组和安全组。
4. 向您的任务执行角色添加`CreateSession`权限。

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

5. 运行你的工作。请注意，您必须使用该S3A方案来访问 S3 Express One Zone 存储桶。

```
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
}'
```

运行任务

配置应用程序后，您可以向应用程序提交作业。本节介绍如何使用 AWS CLI 来运行这些作业。本节还确定了 EMR Serverless 上可用的每种应用程序类型的默认值。

主题

- 任务运行状态
- 从 EMR Studio 控制台运行作业
- 正在运行来自的作业 AWS CLI
- 使用经过洗牌优化的磁盘
- 直播职位
- Spark 职位
- Hive 职位
- EMR无服务器 Job 弹性
- 元数据仓配置
- 访问另一个 S3 中的数据 AWS 来自EMR无服务器的账户
- 对EMR无服务器中的错误进行故障排除

任务运行状态

当您将任务运行提交到 Amazon EMR Serverless 任务队列时，该任务运行将进入SUBMITTED状态。任务状态RUNNING一直持续SUBMITTED到达到FAILEDSUCCESS、或CANCELLING。

任务运行可具有以下状态：

状态	描述
已提交	将作业运行提交到 EMR Serverless 时的初始作业状态。该作业正在等待为应用程序安排时间。EMRServerless 开始确定任务运行的优先顺序和计划。
待处理	调度程序正在评估作业运行，以确定应用程序运行的优先级和计划。

状态	描述
已计划	EMRServerless 已为应用程序安排了作业运行，并且正在分配资源来执行该作业。
正在运行	EMRServerless 已分配了作业最初需要的资源，并且该作业正在应用程序中运行。在 Spark 应用程序中，这意味着 Spark 驱动程序进程处于 `running` 状态。
已失败	EMRServerless 未能将作业运行提交给应用程序，或者任务未成功完成。`StateDetails` 有关此任务失败的更多信息，请参阅。
成功	作业运行已成功完成。
正在取消	`CancelJobRun` API已请求取消作业运行，或者作业运行已超时。EMRServerless 正在尝试取消应用程序中的任务并释放资源。
已取消	作业运行已成功取消，其使用的资源已被释放。

从 EMR Studio 控制台运行作业

您可以向EMR无服务器应用程序提交作业运行并从 EMR Studio 控制台查看作业。要在 EMR Studio 控制台上创建或导航到您的EMR无服务器应用程序，请按照控制台入门中的说明进行操作。

提交作业

在提交作业页面上，您可以按如下方式向EMR无服务器应用程序提交作业。

Spark

1. 在名称字段中，输入作业运行的名称。
2. 在 “运行时角色” 字段中，输入您的EMR无服务器应用程序可以在作业运行中IAM扮演的角色的名称。要了解有关运行时角色的更多信息，请参阅Amazon EMR Serverless 的 Job 运行时角色。

3. 在脚本位置字段中，输入脚本或您要运行的脚本JAR的 Amazon S3 位置。对于 Spark 作业，脚本可以是 Python (`.py`) 文件或 JAR (`.jar`) 文件。
4. 如果您的脚本位置是JAR文件，请在主类字段中输入作为作业入口点的类名。
5. （可选）为其余字段输入值。

   - 脚本参数-输入要传递给主脚本JAR或 Python 脚本的任何参数。您的代码会读取这些参数。用逗号分隔数组中的每个参数。
   - Spark 属性-展开 Spark 属性部分，然后在此字段中输入任何 Spark 配置参数。

     > **Note**
     >
     > 如果指定 Spark 驱动程序和执行程序的大小，则必须将内存开销考虑在内。在属`spark.driver.memoryOverhead`性和中指定内存开销值`spark.executor.memoryOverhead`。内存开销的默认值为容器内存的 10%，最小值为 384 MB。执行器内存和内存开销加在一起不能超过工作程序的内存。例如，一个 30 GB 的工作器`spark.executor.memory`上的最大容量必须为 27 GB。

   - Job 配置-在此字段中指定任何作业配置。您可以使用这些作业配置来覆盖应用程序的默认配置。
   - 其他设置-激活或停用 AWS 将 Glue 数据目录粘贴为元数据仓并修改应用程序日志设置。要了解有关元数据仓配置的更多信息，请参阅元数据仓配置。要了解有关应用程序日志记录选项的更多信息，请参阅存储日志。
   - 标签-为应用程序分配自定义标签。
6. 选择提交作业。

Hive

1. 在名称字段中，输入作业运行的名称。
2. 在 “运行时角色” 字段中，输入您的EMR无服务器应用程序可以在作业运行中IAM扮演的角色的名称。
3. 在脚本位置字段中，输入脚本或您要运行的脚本JAR的 Amazon S3 位置。对于 Hive 作业，脚本必须是 Hive (`.sql`) 文件。
4. （可选）为其余字段输入值。

   - 初始化脚本位置-输入在 Hive 脚本运行之前初始化表的脚本的位置。

- Hive 属性-展开 Hive 属性部分，然后在此字段中输入任何 Hive 配置参数。
- Job 配置-指定任何作业配置。您可以使用这些作业配置来覆盖应用程序的默认配置。对于 Hive 作业，`hive.exec.scratchdir`和`hive.metastore.warehouse.dir`是`hive-site`配置中的必需属性。

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

- 其他设置-激活或停用 AWS 将 Glue 数据目录粘贴为元数据仓并修改应用程序日志设置。要了解有关元数据仓配置的更多信息，请参阅元数据仓配置。要了解有关应用程序日志记录选项的更多信息，请参阅存储日志。
- 标签-为应用程序分配任何自定义标签。

5. 选择提交作业。

查看作业运行

在应用程序详细信息页面上的 J ob runs 选项卡中，您可以查看作业运行并对作业运行执行以下操作。

取消作业-要取消处于该`RUNNING`状态的作业运行，请选择此选项。要了解有关任务运行过渡的更多信息，请参阅任务运行状态。

克隆作业-要克隆之前运行的作业并重新提交，请选择此选项。

正在运行来自的作业 AWS CLI

您可以在上创建、描述和删除单个作业 AWS CLI。 您还可以列出所有作业，以便一目了然地查看。

要提交新作业，请使用`start-job-run`。提供要运行的应用程序的 ID 以及特定于作业的属性。有关 Spark 的示例，请参阅Spark 职位。有关 Hive 示例，请参阅Hive 职位。此命令返回您的`application-id`ARN、和新`job-id`的。

每次作业运行都有设定的超时持续时间。如果作业运行超过此持续时间，EMRServerless 将自动取消该任务。默认超时时间为 12 小时。开始运行作业时，可以将此超时设置配置为符合任务要求的值。使用`executionTimeoutMinutes`属性配置该值。

```
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://DOC-EXAMPLE-BUCKET/scripts/create_table.sql",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://DOC-EXAMPLE-BUCKET/hive/
scratch --hiveconf hive.metastore.warehouse.dir=s3://DOC-EXAMPLE-BUCKET/hive/warehouse"
    }
   }' \
  --configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "hive-site",
        "properties": {
            "hive.client.cores": "2",
            "hive.client.memory": "4GIB"
        }
    }]
}'
```

要描述工作，请使用`get-job-run`。此命令返回特定于作业的配置和新作业的设置容量。

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

要列出您的职位，请使用`list-job-runs`。此命令返回一组简短的属性，其中包括作业类型、状态和其他高级属性。如果您不想查看所有作业，则可以指定要查看的最大作业数，最多 50 个。以下示例指定您要查看最后两次作业的运行情况。

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

```
--application-id application-id
```

要取消任务，请使用cancel-job-run。提供您要取消job-id的任务的 “application-id和”。

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

有关如何运行作业的更多信息，请访问 AWS CLI，请参阅《EMR无服务器API参考》。

使用经过洗牌优化的磁盘

在 Amazon 7.1.0 及更高EMR版本中，您可以在运行 Apache Spark 或 Hive 作业时使用经过洗牌优化的磁盘来提高 I/O 密集型工作负载的性能。与标准磁盘相比，经过洗牌优化的磁盘可提供更高的性能 IOPS（每秒 I/O 操作数），从而加快数据移动速度并减少洗牌操作期间的延迟。Shuffle 优化的磁盘允许您为每个工作人员连接高达 2 TB 的磁盘，因此您可以根据工作负载要求配置适当的容量。

主要优势

Shuffle 优化的磁盘具有以下优点。

- 高IOPS性能 — 经过洗牌优化的磁盘提供的磁盘IOPS比标准磁盘更高，从而在 Spark 和 Hive 作业以及其他洗牌密集型工作负载期间更高效、更快速地进行数据洗牌。
- 更大的磁盘大小 — Shuffle 优化的磁盘支持每位工作人员从 20GB 到 2TB 不等的磁盘大小，因此您可以根据工作负载选择合适的容量。

开始使用

要在工作流程中使用经过洗牌优化的磁盘，请参阅以下步骤。

Spark

1. 使用以下EMR命令创建无服务器 7.1.0 版应用程序。

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

```
  --region <AWS_REGION>
```

2. 将 Spark 作业配置为包含参数`spark.emr-serverless.driver.disk.type`和/或使用`spark.emr-serverless.executor.disk.type`经过洗牌优化的磁盘运行。根据您的用例，您可以使用一个或两个参数。

```
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"
        }
    }'
```

有关更多信息，请参阅 Spark 作业属性。

Hive

1. 使用以下EMR命令创建无服务器 7.1.0 版应用程序。

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

2. 将 Hive 作业配置为包含参数`hive.driver.disk.type`和/或`hive.tez.disk.type`使用经过洗牌优化的磁盘运行。根据您的用例，您可以使用一个或两个参数。

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

有关更多信息，请参阅 Hive 作业属性。

为应用程序配置预先初始化的容量

要创建基于 Amazon EMR 版本 7.1.0 的应用程序，请参阅以下示例。这些应用程序具有以下属性：

- 5 个预初始化的 Spark 驱动程序，每个驱动程序都有 2 v CPU、4 GB 的内存和 50 GB 的洗牌优化磁盘。
- 50 个预初始化的执行器，每个执行器都有 4 v CPU、8 GB 的内存和 500 GB 的洗牌优化磁盘。

当此应用程序运行 Spark 作业时，它会先消耗预先初始化的工作程序，然后将按需工作器扩展到最大容量 400 v CPU 和 1024 GB 内存。或者，您可以省略DRIVER或EXECUTOR的容量。

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"
}'
```

直播职位

EMRServerless 中的流式处理作业是一种任务模式，可让您近乎实时地分析和处理流数据。这些长期运行的作业会轮询流数据，并在数据到达时持续处理结果。流媒体作业最适合需要实时数据处理的任务，例如近实时分析、欺诈检测和推荐引擎。EMR无服务器流式处理作业提供优化，例如内置的作业弹性、实时监控、增强的日志管理以及与流媒体连接器的集成。

以下是流媒体作业的一些用例：

- 近乎实时的分析 — Amazon EMR Serverless 中的流式作业可让您近乎实时地处理流数据，因此您可以对连续的数据流（例如日志数据、传感器数据或点击流数据）进行实时分析，从而根据最新信息得出见解并及时做出决策。
- 欺诈检测 — 当您分析数据流并识别出现的可疑模式或异常时，您可以使用流式作业在金融交易、信用卡操作或在线活动中进行近乎实时的欺诈检测。
- 推荐引擎 — 流式作业可以处理用户活动数据并更新推荐模型。这样做可以根据行为和偏好提供个性化和实时的推荐。
- 社交媒体分析 — 流媒体作业可以处理社交媒体数据，例如推文、评论和帖子，因此组织可以近乎实时地监控趋势、情绪分析和管理品牌声誉。
- 物联网 (IoT) 分析 — 流式作业可以处理和分析来自物联网设备、传感器和互联机器的高速数据流，因此您可以运行异常检测、预测性维护和其他物联网分析用例。
- 点击流分析 — 流式作业可以处理和分析来自网站或移动应用程序的点击流数据。使用此类数据的企业可以进行分析，以进一步了解用户行为、个性化用户体验并优化营销活动。

- 日志监控和分析 — 流式作业还可以处理来自服务器、应用程序和网络设备的日志数据。这为您提供了异常检测、故障排除以及系统运行状况和性能。

主要好处

EMRServerless 中的流式作业会自动提供工作弹性，这是以下因素的组合：

- 自动重试 — EMR Serverless 会自动重试任何失败的作业，而无需您手动输入。
- 可用区 (AZ) 弹性 — 如果原始可用区遇到问题，EMRServerless 会自动将流媒体作业切换到健康的可用区。
- 日志管理：
  - 日志轮换 — 为了更有效地管理磁盘存储，EMRServerless 会定期轮换长时间流式处理作业的日志。这样做可以防止可能占用所有磁盘空间的日志积累。
  - 日志压缩-帮助您在托管持久性中高效管理和优化日志文件。当您使用托管 Spark 历史服务器时，Compaction 还可以改善调试体验。

支持的数据源和数据接收器

EMRServerless 可与许多输入数据源和输出数据接收器配合使用：

- 支持的输入数据源 — 亚马逊 Kinesis Data Streams、适用于 Apache Kafka 的亚马逊托管流媒体和自我管理的 Apache Kafka 集群。默认情况下，亚马逊 7.1.0 及更高EMR版本包含亚马逊 Kinesis Data Stream s 连接器，因此您无需构建或下载任何其他软件包。
- 支持的输出数据接收器 — AWS Glue 数据目录表、亚马逊 S3、亚马逊 Redshift、My、Postg SQL re Oracle SQL、甲骨文、微软、Apache Iceberg SQL、Delta Lake 和 Apache Hudi。

注意事项和限制

使用流式传输作业时，请记住以下注意事项和限制。

- Amazon 7.1.0 及更高EMR版本支持直播作业。
- EMRServerless 预计流式处理作业会运行很长时间，因此您无法通过设置执行超时来限制作业的运行时间。
- 流媒体作业仅与 Spark 引擎兼容，后者建立在结构化流媒体框架之上。
- EMRServerless 会无限期地重试流媒体作业，并且您无法自定义最大尝试次数。如果失败的尝试次数超过了每小时窗口内设置的阈值，则会自动包括Thrash Prevention，以停止作业重试。默认阈值

为一小时内五次失败尝试。您可以将此阈值配置为 1 到 10 次尝试之间。有关更多信息，请参阅 Job 弹性。

- 流式处理作业具有用于保存运行时状态和进度的检查点，因此 EMR Serverless 可以从最新的检查点恢复流式处理作业。有关更多信息，请参阅 Apache Spark 文档中的使用检查点从故障中恢复。

开始直播作业

请参阅以下说明，了解如何开始使用流媒体作业。

1. 按照 Amazon EMR Serverless 入门指南创建应用程序。请注意，您的应用程序必须运行亚马逊 7.1.0 或更高EMR版本。
2. 应用程序准备就绪后，将mode参数设置为STREAMING以提交流媒体作业，如下所示 AWS CLI 示例。

```
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"
    }
}'
```

支持的流媒体连接器

流连接器便于从流媒体源读取数据，还可以将数据写入流接收器。

以下是支持的流媒体连接器：

亚马逊 Kinesis Data Streams 连接器

适用于 Apache Spark 的 Amazon Kinesis Data Stream s 连接器支持构建流应用程序和管道，这些应用程序和管道使用来自亚马逊 Kinesis Data Streams 的数据并向其写入数据。该连接器支持增强的扇出消耗，每个分片的专用读取吞吐率高达 2MB/秒。默认情况下，Amazon EMR Serverless 7.1.0 及更高版本包含连接器，因此您无需构建或下载任何其他软件包。有关连接器的更多信息，请参阅上的 spark-sql-kinesis-connector 页面 GitHub。

以下是如何使用 Kinesis Data Streams 连接器依赖项启动作业运行的示例。

```
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"
    }
}'
```

要连接到 Kinesis Dat VPC a Streams，您必须为无服务器应用程序VPC配置EMR访问权限，并使用端点允许私有访问。或者使用网关获取NAT公共访问权限。有关更多信息，请参阅配置VPC访问权限。您还必须确保您的作业运行时角色具有访问所需数据流所需的读取和写入权限。要详细了解如何配置作业运行时角色，请参阅 Amazon EMR Serverless 的作业运行时角色。有关所有必需权限的完整列表，请参阅上的spark-sql-kinesis-connector 页面 GitHub。

阿帕奇 Kafka 连接器

适用于 Spark 结构化流媒体的 Apache Kafka 连接器是 Spark 社区提供的开源连接器，可在 Maven 存储库中使用。此连接器便于 Spark 结构化流媒体应用程序从自我管理的 Apache Kafka 和适用于 Apache Kafka 的 Apache Managed Streaming 读取数据并向其写入数据。有关连接器的更多信息，请参阅 Apache Spark 文档中的结构化流媒体 + Kafka 集成指南。

以下示例演示了如何在任务运行请求中包含 Kafka 连接器。

```
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 连接器版本取决于你的EMR无服务器发布版本和相应的 Spark 版本。要找到正确的 Kafka 版本，请参阅结构化直播 + Kafka 集成指南。

要使用IAM带身份验证的 Apache Kafka 版亚马逊托管流媒体，您必须添加另一个依赖项才能让 Kafka 连接器连接到亚马逊。MSK IAM有关更多信息，请参阅上的aws-msk-iam-auth 存储库 GitHub。您还必须确保作业运行时角色具有必要的IAM权限。以下示例演示如何使用带IAM身份验证的连接器。

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

要使用来自 Amazon 的 Kafka 连接器和IAM身份验证库，MSK您必须为EMR无服务器应用程序配置访问权限。VPC您的子网必须具有互联网访问权限并使用NAT网关才能访问 Maven 依赖项。有关更多信息，请参阅配置VPC访问权限。子网必须具有网络连接才能访问 Kafka 集群。无论你的 Kafka 集群是自行管理的，还是使用适用于 Apache Kafka 的亚马逊托管流媒体，都是如此。

直播作业日志管理

日志轮换

流式处理作业支持 Spark 应用程序日志和事件日志的日志轮换。日志轮换可防止长时间流式传输作业生成可能占用所有可用磁盘空间的大型日志文件。日志轮换可帮助您节省磁盘存储空间并防止由于磁盘空间不足而导致任务失败。有关更多信息，请参阅轮换日志。

原木压实

每当有托管日志可用时，流式处理作业还支持对 Spark 事件日志进行日志压缩。有关托管日志记录的更多详细信息，请参阅使用托管存储进行日志记录。流式处理作业可以运行很长时间，事件数据量会随着时间的推移而积累，从而显著增加日志文件的大小。Spark 历史服务器读取这些事件并将其加载到 Spark 应用程序用户界面的内存中。此过程可能会导致高延迟和高成本，尤其是在存储在 Amazon S3 中的事件日志非常大的情况下。

日志压缩可减小事件日志的大小，因此 Spark History Server 在任何时候都不必加载超过 1 GB 的事件日志。有关更多信息，请参阅 Apache Spark 文档中的监控和检测。

Spark 职位

您可以在`type`参数设置为的情况下在应用程序上运行 Spark 作业`SPARK`。任务必须与与亚马逊EMR发行版兼容的 Spark 版本兼容。例如，当你使用亚马逊EMR版本 6.6.0 运行作业时，你的任务必须与 Apache Spark 3.2.0 兼容。有关每个版本的应用程序版本的信息，请参阅Amazon EMR 无服务器发布版本。

Spark 作业参数

使用运行 Spark 作业时，可以指定以下参数。`StartJobRun`API

必需参数

- Spark 作业运行时角色
- Spark 作业驱动程序参数

- Spark 配置覆盖参数
- Spark 动态资源分配优化

Spark 作业运行时角色

用于**executionRoleArn**为应用程序ARN用于执行 Spark 作业的IAM角色指定。此角色必须包含以下权限：

- 从您的数据所在的 S3 存储桶或其他数据源中读取
- 从 PySpark 脚本或文件所在的 S3 存储桶或前缀中读取 JAR
- 写入您打算在其中写入最终输出的 S3 存储桶
- 将日志写入`S3MonitoringConfiguration`指定的 S3 存储桶或前缀
- 如果您使用KMS密钥加密 S3 存储桶中的数据，则可以访问KMS密钥
- 访问 AWS 如果你使用 Spark，则使用 Glue 数据目录 SQL

如果您的 Spark 作业向其他数据源读取数据或从其他数据源写入数据，请在此IAM角色中指定相应的权限。如果您不向IAM角色提供这些权限，则任务可能会失败。有关更多信息，请参阅Amazon EMR Serverless 的 Job 运行时角色 和存储日志。

Spark 作业驱动程序参数

用于**jobDriver**为作业提供输入。对于要运行的作业类型，作业驱动程序参数仅接受一个值。对于 Spark 作业，参数值为`sparkSubmit`。您可以使用此作业类型通过 Spark 提交来运行 Scala、Java PySpark、、SparkR 以及任何其他支持的作业。Spark 作业具有以下参数：

- **sparkSubmitParameters**— 这些是您要发送到任务的其他 Spark 参数。使用此参数可以覆盖默认的 Spark 属性，例如驱动程序内存或执行器数量，例如`--conf`或`--class`参数中定义的属性。
- **entryPointArguments**— 这是您要传递给主文件JAR或 Python 文件的一组参数。您应该使用 Entrypoint 代码来处理读取这些参数。用逗号分隔数组中的每个参数。
- **entryPoint**— 这是 Amazon S3 中对你要运行的主文件JAR或 Python 文件的引用。如果您运行的是 Scala 或 JavaJAR，请`SparkSubmitParameters`使用`--class`参数在中指定主入口类。

有关更多信息，请参阅使用 spark-submit 启动应用程序。

Spark 配置覆盖参数

configurationOverrides用于覆盖监控级别和应用程序级别的配置属性。此参数接受具有以下两个字段的JSON对象：

- **monitoringConfiguration**-使用此字段指定您希望EMR无服务器作业存储您的 Spark 任务日志的 Amazon S3 URL (s3MonitoringConfiguration)。请确保您使用相同的存储桶创建了此存储桶 AWS 账户 它托管你的应用程序，而且在同一个地方 AWS 区域 你的任务在哪里运行。
- **applicationConfiguration**— 要覆盖应用程序的默认配置，可以在此字段中提供配置对象。您可以使用速记语法来提供配置，也可以在文件中引用配置对象。JSON配置对象包含分类、属性和可选的嵌套配置。属性由您希望在该文件中覆盖的设置组成。您可以在单个JSON对象中为多个应用程序指定多个分类。

> **Note**
>
> 可用的配置分类因特定的EMR无服务器版本而异。例如，自定义 Log4j `spark-driver-log4j2` 和的分类仅`spark-executor-log4j2`在 6.8.0 及更高版本中可用。

如果您在应用程序覆盖和 Spark 提交参数中使用相同的配置，则 Spark 提交参数优先。配置的优先级按以下顺序排列，从高到低：

- EMRServerless 在创建`SparkSession`时提供的配置。
- 您在`--conf`参数中`sparkSubmitParameters`提供的配置。
- 当你启动作业时，你作为应用程序的一部分提供的配置会被覆盖。
- 您在创建应用程序`runtimeConfiguration`时作为一部分提供的配置。
- 优化了 Amazon 在该版本中EMR使用的配置。
- 应用程序的默认开源配置。

有关在应用程序级别声明配置以及在作业运行期间覆盖配置的更多信息，请参阅EMR无服务器的默认应用程序配置。

Spark 动态资源分配优化

`dynamicAllocationOptimization`用于优化EMR无服务器中的资源使用情况。`true`在你的 Spark 配置分类中将此属性设置为表示EMR无服务器需要优化执行器资源分配，以便更好地将 Spark 请求和取消执行者的速率与 S EMR erverless 创建和释放工作人员的速率保持一致。通过这样

做，EMRServerless 可以更优化地跨阶段重复使用工作人员，从而在运行多个阶段的作业时降低成本，同时保持相同的性能。

此属性在所有 Amazon EMR 发行版本中都可用。

以下是配置分类的示例dynamicAllocationOptimization。

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

如果您使用的是动态分配优化，请考虑以下几点：

- 此优化适用于您为其启用了动态资源分配的 Spark 作业。
- 为了实现最佳成本效益，我们建议根据您的工作负载使用作业级别设置spark.dynamicAllocation.maxExecutors或应用程序级最大容量设置，为工作人员配置扩展上限。
- 在更简单的工作中，你可能看不到成本的提高。例如，如果您的作业在小型数据集上运行或在一个阶段完成运行，则 Spark 可能不需要更多的执行器或多个扩展事件。
- 具有大阶段、较小阶段然后又是大阶段的作业可能会在作业运行时出现回归。随着 EMR Serverless 更有效地使用资源，这可能会导致较大阶段的可用工作人员减少，从而延长运行时间。

Spark 作业属性

下表列出了可选的 Spark 属性及其默认值，您可以在提交 Spark 作业时覆盖这些属性。

键	描述	默认值
spark.archives	以逗号分隔的档案列表，Spark 将其提取到每个执行者的工作目录中。支持的文件类型包括.jar .tar.gz、.tgz和.zip。要	NULL

键	描述	默认值
	指定要提取的目录名，请在要提取的文件名#后面添加。例如，`file.zip#directory`。	
`spark.authenticate`	用于开启 Spark 内部连接身份验证的选项。	`TRUE`
`spark.driver.cores`	驱动程序使用的内核数。	`4`
`spark.driver.extraJavaOptions`	Spark 驱动程序的额外 Java 选项。	`NULL`
`spark.driver.memory`	驱动程序使用的内存量。	`14G`
`spark.dynamicAllocation.enabled`	开启动态资源分配的选项。此选项可根据工作负载增加或缩小在应用程序中注册的执行者数量。	`TRUE`
`spark.dynamicAllocation.executorIdleTimeout`	在 Spark 将其移除之前，执行者可以保持空闲状态的时间长度。这仅在您开启动态分配时适用。	`60`
`spark.dynamicAllocation.initialExecutors`	开启动态分配时要运行的初始执行者数量。	`3`
`spark.dynamicAllocation.maxExecutors`	如果您启用动态分配，则执行者数量的上限。	对于 6.10.0 及更高版本，`infinity` 对于 6.9.0 及更低版本，`100`
`spark.dynamicAllocation.minExecutors`	如果您启用动态分配，则执行者数量的下限。	`0`

键	描述	默认值
spark.emr-serverless.allocation.batch.size	在每个执行器分配周期中要请求的容器数量。每个分配周期之间有一秒钟的间隔。	20
spark.emr-serverless.driver.disk	Spark 驱动程序磁盘。	20G
spark.emr-serverless.driverEnv. *[KEY]*	将环境变量添加到 Spark 驱动程序的选项。	NULL
spark.emr-serverless.executor.disk	Spark 执行器磁盘。	20G
spark.emr-serverless.memoryOverheadFactor	设置要添加到驱动程序和执行器容器内存中的内存开销。	0.1
spark.emr-serverless.driver.disk.type	连接到 Spark 驱动程序的磁盘类型。	Standard
spark.emr-serverless.executor.disk.type	连接到 Spark 执行程序的磁盘类型。	Standard
spark.executor.cores	每个执行器使用的内核数。	4
spark.executor.extraJavaOptions	Spark 执行器的额外 Java 选项。	NULL
spark.executor.instances	要分配的 Spark 执行器容器的数量。	3
spark.executor.memory	每个执行器使用的内存量。	14G
spark.executorEnv. *[KEY]*	向 Spark 执行器添加环境变量的选项。	NULL

键	描述	默认值
spark.files	每个执行器工作目录中要存放的文件列表，以逗号分隔。您可以使用访问执行器中这些文件的文件路径。SparkFiles.get(*fileName*)	NULL
spark.hadoop.hive.metastore.client.factory.class	Hive 元数据仓实现类。	NULL
spark.jars	要添加到驱动程序和执行程序的运行时类路径中的其他 jar。	NULL
spark.network.crypto.enabled	开启AES基于RPC加密的选项。这包括在 Spark 2.2.0 中添加的身份验证协议。	FALSE
spark.sql.warehouse.dir	托管数据库和表的默认位置。	$PWD/spark-warehouse
spark.submit.pyFiles	要放置在 PYTHONPATH Python 应用程序中的.zip.egg、或 .py文件的逗号分隔列表。	NULL

下表列出了默认的 Spark 提交参数。

键	描述	默认值
archives	以逗号分隔的档案列表，Spark 将其提取到每个执行者的工作目录中。	NULL
class	应用程序的主类（适用于 Java 和 Scala 应用程序）。	NULL

键	描述	默认值
conf	一个任意 Spark 配置属性。	NULL
driver-cores	驱动程序使用的内核数。	4
driver-memory	驱动程序使用的内存量。	14G
executor-cores	每个执行器使用的内核数。	4
executor-memory	执行程序使用的内存量。	14G
files	要放置在每个执行器工作目录中的文件列表，以逗号分隔。您可以使用访问执行器中这些文件的文件路径。SparkFiles.get(*fileName*)	NULL
jars	要包含在驱动程序和执行器类路径中的 jar 的逗号分隔列表。	NULL
num-executors	要启动的执行者数量。	3
py-files	要放置在 PYTHONPATH Python 应用程序上的.zip.egg、或.py文件的逗号分隔列表。	NULL
verbose	开启其他调试输出的选项。	NULL

火花示例

以下示例说明如何使用运行 Python 脚本。StartJobRun API有关使用此示例的 end-to-end 教程，请参阅开始使用 Amazon EMR Serverless。您可以在EMR无服务器示例 GitHub 存储库中找到有关如何运行 PySpark 作业和添加 Python 依赖关系的其他示例。

```
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://DOC-EXAMPLE-BUCKET-OUTPUT/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"
        }
    }'
```

以下示例说明如何使用运行 Spark JAR。StartJobRun API

```
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 职位

您可以在type参数设置为的情况下在应用程序上运行 Hive 作业。HIVE任务必须与与 Amazon EMR 发行版兼容的 Hive 版本兼容。例如，当您在亚马逊EMR版本 6.6.0 的应用程序上运行作业时，您的任务必须与 Apache Hive 3.1.2 兼容。有关每个版本的应用程序版本的信息，请参阅Amazon EMR 无服务器发布版本。

Hive 作业参数

使用运行 Hive 作业时，必须指定以下参数。StartJobRunAPI

必需参数

- Hive 作业运行时角色
- Hive 作业驱动程序参数

- Hive 配置覆盖参数

Hive 作业运行时角色

用于**executionRoleArn**为应用程序ARN用于执行 Hive 作业的IAM角色指定。此角色必须包含以下权限：

- 从您的数据所在的 S3 存储桶或其他数据源中读取
- 从 Hive 查询文件和 init 查询文件所在的 S3 存储桶或前缀中读取
- 读取并写入你的 Hive Scratch 目录和 Hive Metastore 仓库目录所在的 S3 存储桶
- 写入您打算在其中写入最终输出的 S3 存储桶
- 将日志写入`S3MonitoringConfiguration`指定的 S3 存储桶或前缀
- 如果您使用KMS密钥加密 S3 存储桶中的数据，则可以访问KMS密钥
- 访问 AWS Glue 数据目录

如果您的 Hive 作业向其他数据源读取或写入数据，请在此IAM角色中指定相应的权限。如果您不向该IAM角色提供这些权限，则您的任务可能会失败。有关更多信息，请参阅 Amazon EMR Serverless 的 Job 运行时角色。

Hive 作业驱动程序参数

用于**jobDriver**为作业提供输入。对于要运行的作业类型，作业驱动程序参数仅接受一个值。当你指定`hive`为作业类型时，EMRServerless 会向参数传递 Hive 查询。`jobDriver`Hive 作业具有以下参数：

- **query**— 这是 Amazon S3 中对你要运行的 Hive 查询文件的引用。
- **parameters**— 这些是您要覆盖的其他 Hive 配置属性。要覆盖属性，请将它们作为传递给此参数`--hiveconf` *`property=value`*。要覆盖变量，请将它们作为传递给此参数`--hivevar` *`key=value`*。
- **initQueryFile**— 这是初始化的 Hive 查询文件。Hive 会在您查询之前运行此文件，并可以使用它来初始化表。

Hive 配置覆盖参数

configurationOverrides用于覆盖监控级别和应用程序级别的配置属性。此参数接受具有以下两个字段的JSON对象：

- **monitoringConfiguration**— 使用此字段指定您希望EMR无服务器作业存储您的 Hive 作业日志的 Amazon S3 URL (s3MonitoringConfiguration)。请务必使用相同的存储桶创建此存储桶 AWS 账户 它托管你的应用程序，而且在同一个地方 AWS 区域 你的任务在哪里运行。
- **applicationConfiguration**— 您可以在此字段中提供配置对象来覆盖应用程序的默认配置。您可以使用速记语法来提供配置，也可以在文件中引用配置对象。JSON配置对象包含分类、属性和可选的嵌套配置。属性由您希望在该文件中覆盖的设置组成。您可以在单个JSON对象中为多个应用程序指定多个分类。

> **Note**
>
> 可用的配置分类因特定的EMR无服务器版本而异。例如，自定义 Log4j spark-driver-log4j2 和的分类仅spark-executor-log4j2在 6.8.0 及更高版本中可用。

如果您在应用程序覆盖和 Hive 参数中传递相同的配置，则 Hive 参数优先。以下列表按优先级从最高优先级到最低优先级对配置进行排名。

- 作为 Hive 参数的一部分提供的配置。--hiveconf *property=value*
- 当你启动作业时，你作为应用程序的一部分提供的配置会被覆盖。
- 您在创建应用程序runtimeConfiguration时作为一部分提供的配置。
- 优化了 Amazon 为该版本EMR分配的配置。
- 应用程序的默认开源配置。

有关在应用程序级别声明配置以及在作业运行期间覆盖配置的更多信息，请参阅EMR无服务器的默认应用程序配置。

Hive 作业属性

下表列出了您在提交 Hive 作业时必须配置的必备属性。

设置	描述
hive.exec.scratchdir	在 Hive 任务执行期间，S EMR erverless 在其中创建临时文件的 Amazon S3 位置。

设置	描述
hive.metastore.warehouse.dir	Hive 中托管表的数据库在 Amazon S3 中的位置。

下表列出了可选的 Hive 属性及其默认值，您可以在提交 Hive 作业时覆盖这些属性。

设置	描述	默认值
fs.s3.customAWSCredentialsProvider	这些区域有： AWS 您要使用的凭证提供商。	com.amazonaws.auth d efaultAWSCredentials ProviderChain
fs.s3a.aws.credentials.provider	这些区域有： AWS 您要在 S3A 文件系统中使用的凭据提供程序。	com.amazonaws.auth d efaultAWSCredentials ProviderChain
hive.auto.convert.join	该选项根据输入文件的大小启用将常用联接自动转换为 mapjoin。	TRUE
hive.auto.convert.join.noconditionaltask	该选项在 Hive 根据输入文件大小将普通联接转换为 mapjoin 时启用优化。	TRUE
hive.auto.convert.join.noconditionaltask.size	连接会直接转换为小于此大小的 mapJoin。	最优值是基于 Tez 任务内存计算得出的
hive.cbo.enable	使用方解石框架启用基于成本的优化的选项。	TRUE
hive.cli.tez.session.async	在编译 Hive 查询时可以选择启动后台 Tez 会话。设置为时false，Tez AM 将在编译 Hive 查询后启动。	TRUE

设置	描述	默认值
hive.compute.query.using.stats	激活 Hive 以使用存储在元存储库中的统计数据来回答某些查询的选项。对于基本统计数据，hive.stats.autogather 请设置为TRUE。要获得更高级的查询集合，请运行analyze table queries。	TRUE
hive.default.fileformat	CREATE TABLE语句的默认文件格式。如果您在CREATE TABLE命令STORED AS [FORMAT]中指定，则可以显式覆盖此设置。	TEXTFILE
hive.driver.cores	用于 Hive 驱动程序进程的内核数。	2
hive.driver.disk	Hive 驱动程序的磁盘大小。	20G
hive.driver.disk.type	Hive 驱动程序的磁盘类型。	Standard
hive.tez.disk.type	tez 工作者的磁盘大小。	Standard
hive.driver.memory	每个 Hive 驱动程序进程要使用的内存量。Hive CLI 和 Tez Application Master 以 20% 的净空平均共享这段内存。	6G
hive.emr-serverless.launch.env.[*KEY*]	可以选择在所有特定于 Hive 的进程（例如 Hive 驱动程序、Tez AM 和 Tez 任务）中设置*KEY*环境变量。	

设置	描述	默认值
hive.exec.dynamic.partition	在DML/中开启动态分区的选项DDL。	TRUE
hive.exec.dynamic.partition.mode	用于指定要使用严格模式还是非严格模式的选项。在严格模式下，必须至少指定一个静态分区，以防意外覆盖所有分区。在非严格模式下，所有分区都允许为动态分区。	strict
hive.exec.max.dynamic.partitions	Hive 总共创建的最大动态分区数。	1000
hive.exec.max.dynamic.partitions.pernode	Hive 在每个映射器和缩减器节点中创建的最大动态分区数。	100
hive.exec.orc.split.strategy	需要以下值之一：BIETL、或HYBRID。这不是用户级配置。 BI指定与查询执行相比，您希望减少在拆分生成上花费的时间。 ETL指定您想在拆分生成中花费更多时间。HYBRID根据启发式方法指定了上述策略的选择。	HYBRID
hive.exec.reducers.bytes.per.reducer	每个减速器的尺寸。默认值为 256 MB。如果输入大小为 1G，则作业使用 4 个减速器。	256000000
hive.exec.reducers.max	减速器的最大数量。	256

设置	描述	默认值
hive.exec.stagingdir	存储 Hive 在表位置和hive.exec.scratchdir 属性中指定的临时目录位置中创建的临时文件的目录名称。	.hive-staging
hive.fetch.task.conversion	需要以下值之一：NONEMINIMAL、或MORE。Hive 可以将选定查询转换为单个FETCH任务。这样可以最大限度地减少延迟。	MORE
hive.groupby.position.alias	使 Hive 在GROUP BY语句中使用列位置别名的选项。	FALSE
hive.input.format	默认输入格式。HiveInputFormat 如果您遇到问题，请将其设置为CombineHiveInputFormat 。	org.apache.hadoop.hive.ql.io.CombineHiveInputFormat
hive.log.explain.output	该选项可启用 Hive 日志中任何查询的扩展输出解释。	FALSE
hive.log.level	Hive 日志级别。	INFO
hive.mapred.reduce.tasks.speculative.execution	为减速器开启投机启动的选项。仅亚马逊 EMR 6.10.x 及更低版本支持。	TRUE
hive.max-task-containers	并发容器的最大数量。将配置的映射器内存乘以该值，以确定计算和任务抢占使用的可用内存。	1000
hive.merge.mapfiles	该选项使小文件在仅限地图的作业结束时合并。	TRUE

设置	描述	默认值
hive.merge.size.per.task	作业结束时合并文件的大小。	256000000
hive.merge.tezfiles	在 Tez DAG 结尾处启用合并小文件的选项。	FALSE
hive.metastore.client.factory.class	生成实现IMetaStoreClient 接口的对象的工厂类的名称。	com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory
hive.metastore.glue.catalogid	如果 AWS Glue Data Catalog 充当元存储库，但在不同的存储中运行 AWS 账户 而不是作业，是 ID AWS 账户 作业在哪里运行。	NULL
hive.metastore.uris	元数据仓客户端用来连接到远程元数据仓的节俭URI工具。	NULL
hive.optimize.ppd	开启谓词下推的选项。	TRUE
hive.optimize.ppd.storage	开启对存储处理程序的谓词下推的选项。	TRUE
hive.orderby.position.alias	使 Hive 在ORDER BY语句中使用列位置别名的选项。	TRUE
hive.prewarm.enabled	为 Tez 开启容器预热功能的选项。	FALSE
hive.prewarm.numcontainers	要为 Tez 预热的容器数量。	10
hive.stats.autogather	该选项使 Hive 在INSERT OVERWRITE 命令执行期间自动收集基本统计信息。	TRUE

设置	描述	默认值
hive.stats.fetch.column.stats	关闭从元数据仓获取列统计信息的选项。当列数较多时，获取列统计信息可能会很昂贵。	FALSE
hive.stats.gather.num.threads	partialscan 和noscan分析命令用于分区表的线程数。这仅适用于实现StatsProvidingRecordReader （比如ORC）的文件格式。	10
hive.strict.checks.cartesian.product	开启严格笛卡尔联接检查的选项。这些检查不允许使用笛卡尔积（交叉连接）。	FALSE
hive.strict.checks.type.safety	该选项开启严格类型安全检查并关闭bigint与string和的比较功能double。	TRUE
hive.support.quoted.identifiers	期望值为NONE或COLUMN。NONE表示标识符中只有字母数字和下划线字符有效。COLUMN意味着列名可以包含任何字符。	COLUMN
hive.tez.auto.reducer.parallelism	开启 Tez 自动减速器并行度功能的选项。Hive 仍然会估计数据大小并设置并行度估计值。Tez 对源顶点的输出大小进行采样，并根据需要在运行时调整估计值。	TRUE
hive.tez.container.size	每个 Tez 任务进程要使用的内存量。	6144
hive.tez.cpu.vcores	每个 Tez 任务要使用的内核数量。	2

设置	描述	默认值
hive.tez.disk.size	每个任务容器的磁盘大小。	20G
hive.tez.input.format	Tez AM 中生成拆分的输入格式。	org.apache.hadoop.hive.ql.io.HiveInputFormat
hive.tez.min.partition.factor	Tez 在开启自动减速器并行度时指定的减速器下限。	0.25
hive.vectorized.execution.enabled	开启向量化查询执行模式的选项。	TRUE
hive.vectorized.execution.reduce.enabled	开启查询执行简化侧的矢量化模式的选项。	TRUE
javax.jdo.option.ConnectionDriverName	JDBC元数据仓的驱动程序类名。	org.apache.derby.jdbc.EmbeddedDriver
javax.jdo.option.ConnectionPassword	与元数据仓数据库关联的密码。	NULL
javax.jdo.option.ConnectionURL	JDBC元数据仓的JDBC连接字符串。	jdbc:derby:;databaseName=metastore_db;create=true
javax.jdo.option.ConnectionUserName	与元数据仓数据库关联的用户名。	NULL
mapreduce.input.fileinputformat.split.maxsize	当您的输入格式为时，拆分计算期间分割的最大大小。org.apache.hadoop.hive.ql.io.CombineHiveInputFormat 值为 0 表示没有限制。	0

设置	描述	默认值
tez.am.dag.cleanup.on.completion	完成后开启清理随机播放数据的选项。DAG	TRUE
tez.am.emr-serverless.launch.env.[*KEY*]	在 Tez AM 流程中设置*KEY*环境变量的选项。对于 Tez AM，此值会覆盖该 hive.emr-serverless.launch.env.[*KEY*]值。	
tez.am.log.level	Serverl EMR ess 传递给 Tez 应用程序主服务器的根日志记录级别。	INFO
tez.am.sleep.time.before.exit.millis	EMRServerless 应在 AM 关闭请求后的这段时间之后推送ATS事件。	0
tez.am.speculation.enabled	导致推测性启动较慢任务的选项。当某些任务由于机器故障或运行缓慢而导致运行速度变慢时，这可以帮助减少作业延迟。仅亚马逊 EMR 6.10.x 及更低版本支持。	FALSE
tez.am.task.max.failed.attempts	在特定任务失败之前，该任务可能失败的最大尝试次数。此数字不计算手动终止的尝试次数。	3
tez.am.vertex.cleanup.height	一个距离，如果所有从属顶点都已完成，Tez AM 将删除顶点随机数据。当值为 0 时，此功能将关闭。Amazon 6.8.0 及更高EMR版本支持此功能。	0

设置	描述	默认值
tez.client.asynchronous-stop	该选项可让 S EMR erverless 在终止 Hive 驱动程序之前推送ATS事件。	FALSE
tez.grouping.max-size	分组拆分的大小上限（以字节为单位）。此限制可防止分割过大。	1073741824
tez.grouping.min-size	分组拆分的大小下限（以字节为单位）。此限制可防止太多的小分割。	16777216
tez.runtime.io.sort.mb	Tez 对输出进行排序时软缓冲区的大小已排序。	最优值是基于 Tez 任务存储器计算得出的
tez.runtime.unordered.output.buffer.size-mb	Tez 不直接写入磁盘时要使用的缓冲区大小。	最优值是基于 Tez 任务存储器计算得出的
tez.shuffle-vertex-manager.max-src-fraction	在 S EMR erverless 为当前顶点（如果是ScatterGather 连接）安排所有任务之前，必须完成的源任务的比例。在当前顶点上准备好调度的任务数量在和之间min-fraction 线性缩放。max-fraction 这默认为默认值或tez.shuffle-vertex-manager.min-src-fraction ，以较大者为准。	0.75
tez.shuffle-vertex-manager.min-src-fraction	在 S EMR erverless 为当前顶点安排任务之前，必须完成的源任务比例（如果是ScatterGather 连接）。	0.25

设置	描述	默认值
tez.task.emr-serverless.launch.env.[*KEY*]	在 Tez 任务流程中设置*KEY*环境变量的选项。对于 Tez 任务，此值会覆盖该hive.emr-serverless.launch.env.[*KEY*]值。	
tez.task.log.level	EMRServerless 传递给 Tez 任务的根日志级别。	INFO
tez.yarn.ats.event.flush.timeout.millis	AM 在关闭之前应等待事件刷新的最长时间。	300000

Hive 工作示例

以下代码示例显示了如何使用运行 Hive 查询。StartJobRun API

```
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"
```

```
            }
        }]
    }'
```

您可以在EMR无服务器示例存储库中找到有关如何运行 Hive 作业的其他示例 GitHub 。

EMR无服务器 Job 弹性

EMRServerless 版本 7.1.0 及更高版本包括对作业弹性的支持，因此它会自动重试任何失败的作业，而无需您手动输入。作业弹性的另一个好处是，如果可用区遇到任何问题，EMRServerless 可以将作业运行转移到不同的可用区 (AZ)。

要为作业启用作业弹性，请为您的作业设置重试策略。重试策略可确保 EMR Serverless 在作业失败时自动重启该作业。批处理和流式处理作业都支持重试策略，因此您可以根据自己的用例自定义作业弹性。下表比较了批处理和流式作业之间作业弹性的行为和差异。

	批处理作业	直播职位
默认行为	不重新运行作业。	始终重试运行作业，因为应用程序在运行作业时会创建检查点。
重试点	Batch 作业没有检查点，因此 EMR Serverless 总是从头开始重新运行作业。	流式处理作业支持检查点，因此您可以配置流式处理查询以保存运行时状态并进度到 Amazon S3 中的检查点位置。EMRServerless 从检查点恢复作业运行。有关更多信息，请参阅 Apache Spark 文档中的使用检查点从故障中恢复。
最大重试次数	最多允许 10 次重试。	流式传输作业具有内置的抖动预防控制功能，因此，如果作业在一小时后继续失败，应用程序将停止重试作业。一小时内的默认重试次数为五次尝试。您可以将此重试次数配置

	批处理作业	直播职位
		为 1 或 10 之间。您无法自定义最大尝试次数。值为 1 表示不重试。

当 EMR Serverless 尝试重新运行作业时，它还会使用尝试次数为该作业编制索引，这样您就可以在尝试中跟踪作业的生命周期。

你可以使用EMR无服务器API操作或 AWS CLI 更改工作弹性或查看与工作弹性相关的信息。有关更多信息，请参阅EMR无服务器API指南。

默认情况下，EMRServerless 不会重新运行批处理作业。要启用批处理作业的重试功能，请在启动批处理作业运行时配置maxAttempts参数。该maxAttempts参数仅适用于批处理作业。默认值为 1，这意味着不重新运行作业。可接受的值为 1 到 10（含）。

以下示例演示了如何在启动作业运行时指定最多 10 次尝试次数。

```
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如果流媒体作业失败，Serverless 会无限期地重试这些作业。要防止由于重复出现不可恢复的故障而导致抖动，请使用为流式处理maxFailedAttemptsPerHour作业重试配置抖动防护控制。此参数允许您指定在 EMR Serverless 停止重试前一小时内允许的最大失败尝试次数。默认值为五。可接受的值为 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"
     }
}'
```

您还可以使用其他作业运行API操作来获取有关作业的信息。例如，您可以将attempt参数与GetJobRun操作配合使用，以获取有关特定任务尝试的详细信息。如果不包含attempt参数，则该操作将返回有关最新尝试的信息。

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

该ListJobRunAttempts操作会返回与作业运行相关的所有尝试的相关信息。

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

该GetDashboardForJobRun操作会创建并返回一个URL，您可以使用它来访问应用程序UIs以运行作业。该attempt参数允许您URL针对特定尝试获取。如果不包含attempt参数，则该操作将返回有关最新尝试的信息。

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

使用重试策略监控作业

作业弹性支持还添加了新的EMR无服务器作业运行重试事件。EMRServerless 会在每次重试作业时发布此事件。您可以使用此通知来跟踪任务的重试次数。有关事件的更多信息，请参阅 Amazon EventBridge 事件。

使用重试策略进行登录

每当 EMR Serverless 重试作业时，该尝试都会生成自己的日志集。为了确保 S EMR erverless 能够在不覆盖任何日志 CloudWatch 的情况下成功将这些日志传输到 Amazon S3 和 Amazon，S EMR erverless 在 S3 日志路径和 CloudWatch 日志流名称的格式中添加了一个前缀，以包含任务的尝试次数。

以下是格式的示例。

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

这种格式可确保 EMR Serverless 将每次尝试执行任务的所有日志发布到自己在 Amazon S3 中的指定位置和 CloudWatch。有关更多详细信息，请参阅存储日志。

> **Note**
>
> EMRServerless 仅对所有流式处理作业和任何启用了重试的批处理作业使用此前缀格式。

元数据仓配置

Hive 元数据仓是一个集中位置，用于存储有关表的结构信息，包括架构、分区名称和数据类型。使用 EMR Serverless，您可以将此表元数据保存在可以访问您的作业的元数据存储中。

Hive 元数据仓有两种选择：

- 这些区域有： AWS Glue 数据目录
- 外部 Apache Hive 元存储库

使用 AWS Glue 数据目录作为元数据库

你可以将你的 Spark 和 Hive 作业配置为使用 AWS Glue 数据目录作为其元数据库。如果您需要永久性元数据仓或由不同的应用程序、服务或共享的元数据仓，我们建议您使用此配置 AWS 账户。 有关

数据目录的更多信息，请参阅填充 AWS Glue 数据目录。有关信息 AWS Glue 定价，见 AWS Glue 定价。

您可以将您的EMR无服务器作业配置为使用 AWS Glue 数据目录要么在同一个目录中 AWS 账户 作为你的应用程序，或者在不同的应用程序中 AWS 账户.

配置 AWS Glue 数据目录

要配置数据目录，请选择要使用的EMR无服务器应用程序类型。

Spark

当你使用 EMR Studio 通过EMR无服务器 Spark 应用程序运行作业时， AWS Glue 数据目录是默认的元数据库。

当你使用SDKs或 AWS CLI，您可以在作业运行的sparkSubmit参数com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory中将spark.hadoop.hive.metastore.client.factory.class配置设置为。以下示例说明如何使用配置数据目录 AWS CLI.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://DOC-EXAMPLE-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，你可以在你的 Spark 代码SparkSession中创建新配置时设置此配置。

```
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

对于EMR无服务器 Hive 应用程序，数据目录是默认的元数据库。也就是说，当您在EMR无服务器 Hive 应用程序上运行作业时，Hive 会在数据目录中以相同的方式记录元存储信息 AWS 账户 作为您的应用程序。您不需要虚拟私有云 (VPC) 即可将数据目录用作元数据仓。

要访问 Hive 元数据仓表，请添加所需的 AWS 设置IAM权限中概述的 Glue 政策 AWS Glue 。

为EMR无服务器配置跨账户访问权限和 AWS Glue 数据目录

要为 EMR Serverless 设置跨账户访问权限，您必须先登录以下网站 AWS 账户:

- AccountA— 一个 AWS 账户 您在其中创建了EMR无服务器应用程序。
- AccountB— 一个 AWS 账户 其中包含一个 AWS 你想让你的EMR无服务器作业运行访问的 Glue 数据目录。

1. 确保中的管理员或其他授权身份将资源策略AccountB附加到中的数据目录AccountB。此策略授予对AccountB目录中的资源执行操作的AccountA特定跨账户权限。

```
{
  "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. 向中的EMR无服务器作业运行时角色添加IAM策略，AccountA以便该角色可以访问中的AccountB数据目录资源。

```
{
  "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. 开始运行作业。根据EMR无服务器应用程序AccountA的类型，此步骤略有不同。

Spark

在hive-site分类中设置该spark.hadoop.hive.metastore.glue.catalogid属性，如以下示例所示。Replace（替换）*## b-catalog-ID* 其中包含数据目录的 ID AccountB。

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

Hive

在hive-site分类中设置该hive.metastore.glue.catalogid属性，如以下示例所示。Replace（替换）*## b-catalog-ID* 其中包含数据目录的 ID AccountB。

```
aws emr-serverless start-job-run \
--application-id "application-id" \
--execution-role-arn "job-role-arn" \
--job-driver '{
    "hive": {
    "query": "s3://DOC-EXAMPLE-BUCKET/hive/scripts/create_table.sql",
    "parameters": "--hiveconf hive.exec.scratchdir=s3://DOC-EXAMPLE-BUCKET/hive/
scratch --hiveconf hive.metastore.warehouse.dir=s3://DOC-EXAMPLE-BUCKET/hive/
warehouse"
```

```
    }
}' \
--configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "hive-site",
        "properties": {
            "hive.metastore.glue.catalogid": "AccountB-catalog-id"
        }
    }]
}'
```

使用时的注意事项 AWS Glue 数据目录

您可以在 Hive 脚本ADD JAR中JARs使用添加辅助工具。有关其他注意事项，请参阅使用时的注意事项 AWS Glue 数据目录。

使用外部 Hive 元数据仓

您可以将EMR无服务器 Spark 和 Hive 任务配置为连接到外部 Hive 元存储库，例如亚马逊 Aurora 或 Amazon for My。RDS SQL本节介绍如何设置 Amazon RDS Hive 元数据仓、配置您的VPC以及如何配置您的EMR无服务器任务以使用外部元数据仓。

创建外部 Hive 元数据仓

1. 按照创建中的说明创建带有私有子网的 Amazon Virtual Priv at e Cloud (AmazonVPC)。VPC
2. 使用您的新 Amazon VPC 和私有子网创建您的EMR无服务器应用程序。当您使用 EMR Serverless 应用程序配置时VPC，它会首先为您指定的每个子网配置一个弹性网络接口。然后，它会将您指定的安全组附加到该网络接口。这为您的应用程序提供了访问控制。有关如何设置的更多详细信息VPC，请参阅配置VPC访问权限。
3. 在亚马逊VPC的私有子网中创建 My SQL 或 Aurora Postgre SQL 数据库。有关如何创建亚马逊RDS数据库的信息，请参阅创建亚马逊RDS数据库实例。
4. 按照修改 A mazon RDS 数据库实例中的步骤，修改 My SQL 或 Aurora 数据库的安全组以允许来自您的EMR无服务器安全组的JDBC连接。为来自您的EMR无服务器RDS安全组的入站流量添加规则。

类型	协议	端口范围	来源
全部 TCP	TCP	3306	emr-serverless-security-group

配置 Spark 选项

使用 JDBC

要将您的EMR无服务器 Spark 应用程序配置为连接到基于 Amazon for My 或 Ama RDS zon A SQL urora My SQL 实例的 Hive 元数据仓，请使用连接。JDBC在作业运行`mariadb-connector-java.jar`的`spark-submit`参数`--jars`中传递 with。

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
  --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://DOC-EXAMPLE-BUCKET/scripts/spark-jdbc.py",
            "sparkSubmitParameters": "--jars s3://DOC-EXAMPLE-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://DOC-EXAMPLE-BUCKET/spark/logs/"
```

```
        }
    }
}'
```

以下代码示例是与亚马逊上的 Hive 元数据仓交互的 Spark 入口点脚本。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()
```

使用旧货服务

您可以将EMR无服务器 Hive 应用程序配置为连接基于 Amazon for My 或 Ama RDS zon A SQL urora My 实例的 Hive 元数据仓。SQL为此，请在现有 Amazon EMR 集群的主节点上运行节俭服务器。如果您已经有一个带有旧服务器的 Amazon EMR 集群，想要使用它来简化您的EMR无服务器任务配置，则此选项非常理想。

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
  --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://DOC-EXAMPLE-BUCKET/thriftscript.py",
            "sparkSubmitParameters": "--jars s3://DOC-EXAMPLE-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://DOC-EXAMPLE-BUCKET/spark/logs/"
        }
    }
}'
```

以下代码示例是一个使用节俭协议连接到 Hive 元存储的入口点脚本 (thriftscript.py)。请注意，需要将该hive.metastore.uris属性设置为从外部 Hive 元数据仓读取。

```
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()
```

配置 Hive 选项

使用 JDBC

如果您想在 Amazon M RDS y SQL 或 Amazon Aurora 实例上指定外部 Hive 数据库位置，则可以覆盖默认的元数据仓配置。

> **Note**
>
> 在 Hive 中，您可以同时对元数据仓表执行多次写入。如果您在两个作业之间共享元数据仓信息，请确保不要同时写入同一个元数据仓表，除非您写入同一个元数据仓表的不同分区。

在`hive-site`分类中设置以下配置以激活外部 Hive 元数据仓。

```
{
    "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"
    }
}
```

使用旧货服务器

您可以将EMR无服务器 Hive 应用程序配置为连接到基于 Amazon for My 或 Ama RDS zon A SQL urora M 的 Hive 元数据仓。ySQLinstance为此，请在现有 Amazon EMR 集群的主节点上运行节俭服务器。如果您已经拥有运行旧服务器的 Amazon EMR 集群，并且想要使用您的EMR无服务器任务配置，则此选项非常理想。

在`hive-site`分类中设置以下配置，以便 EMR Serverless 可以访问远程旧货元数据仓。请注意，必须将该`hive.metastore.uris`属性设置为从外部 Hive 元数据仓读取。

```
{
    "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"
    }
}
```

使用外部元数据仓时的注意事项

- 您可以将与 MariaDB 兼容的数据库配置JDBC为元数据仓。这些数据库的示例有 RDS MariaDB、My 和 Amazon A SQL urora。
- 元存储不会自动初始化。如果您的元数据仓未使用 Hive 版本的架构进行初始化，请使用 Hive 架构工具。
- EMR无服务器不支持 Kerberos 身份验证。你不能将具有 Kerberos 身份验证的旧版元数据仓服务器与 Serverl EMR ess Spark 或 Hive 作业一起使用。

访问另一个 S3 中的数据 AWS 来自EMR无服务器的账户

您可以从一个服务器运行 Amazon EMR 无服务器作业 AWS 账户并将它们配置为访问属于另一个存储桶的 Amazon S3 存储桶中的数据 AWS account。本页介绍如何配置从 S EMR erverless 对 S3 的跨账户访问权限。

在 S EMR erverless 上运行的作业可以使用 S3 存储桶策略或代入角色从其他服务器访问 Amazon S3 中的数据 AWS account。

先决条件

要为 Amazon EMR Serverless 设置跨账户访问权限，您必须在登录后完成任务 AWS 账户：

- **`AccountA`**— 这是 AWS 您在其中创建 Amazon EMR 无服务器应用程序的账户。在设置跨账户访问权限之前，您必须在此账户中准备好以下内容：
  - 您要在其中运行作业的 Amazon EMR 无服务器应用程序。
  - 具有在应用程序中运行作业所需的权限的任务执行角色。有关更多信息，请参阅 Amazon EMR Serverless 的 Job 运行时角色。
- **`AccountB`**— 这是 AWS 包含您希望 Amazon EMR 无服务器任务访问的 S3 存储桶的账户。

使用 S3 存储桶策略访问跨账户 S3 数据

要访问 S3 存储桶 account B from account A，将以下策略附加到中的 S3 存储桶 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/*"
         ]
      }
   ]
}
```

有关使用 S3 存储桶策略进行 S3 跨账户访问的更多信息，请参阅 A mazon 简单存储服务用户指南中的示例 2：授予跨账户存储桶权限的存储桶拥有者。

使用代入的角色访问跨账户 S3 数据

为 Amazon EMR Serverless 设置跨账户访问权限的另一种方法是使用中的`AssumeRole`操作 AWS Security Token Service (AWS STS). AWS STS 是一项全球 Web 服务，允许您为用户申请临时的、权限有限的证书。您可以使用自己创建的临时安全证书API调用 S EMR erverless 和 Amazon S3。`AssumeRole`

以下步骤说明如何使用代入的角色从 S EMR erverless 访问跨账户 S3 数据：

1. 创建 Amazon S3 存储桶，*cross-account-bucket*，在AccountB。有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的创建存储桶。如果您希望跨账户访问 DynamoDB，还可以在 AccountB 中创建 DynamoDB 表。有关更多信息，请参阅亚马逊 DynamoDB 开发者指南中的创建 Dy namoD B 表。
2. 在中创建一个AccountB可以访问的Cross-Account-Role-BIAM角色 *cross-account-bucket*.
   a. 登录 AWS Management Console 然后打开IAM控制台，网址为https://console.aws.amazon.com/iam/。
   b. 选择 Roles (角色) 并创建新角色：Cross-Account-Role-B。有关如何创建IAM角色的更多信息，请参阅《IAM用户指南》中的创建IAM角色。
   c. 创建一个IAM策略，指定访问权限Cross-Account-Role-B的策略 *cross-account-bucket* S3 存储桶，如以下策略声明所示。然后将该IAM策略附加到Cross-Account-Role-B。有关更多信息，请参阅《IAM用户指南》中的创建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 访问权限，请IAM创建一个策略来指定访问跨账户 DynamoDB 表的权限。然后将该IAM策略附加到Cross-Account-Role-B。有关更多信息，请参阅《用户指南》中的IAM Amazon DynamoDB：允许访问特定表。

以下是允许访问 DynamoDB 表的策略。CrossAccountTable

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

```
            "Action": "dynamodb:*",
            "Resource": "arn:aws:dynamodb:MyRegion:AccountB:table/CrossAccountTable"
        }
    ]
}
```

3. 编辑 Cross-Account-Role-B 角色的信任关系。
   a. 要为角色配置信任关系，请在IAM控制台中为在步骤 2 中创建Cross-Account-Role-B的角色选择信任关系选项卡。
   b. 选择 Edit Trust Relationship (编辑信任关系)。
   c. 添加以下政策文档。这AccountA允许Job-Execution-Role-A他们Cross-Account-Role-B扮演这个角色。

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

4. 格兰特Job-Execution-Role-AAccountA在 AWS STS AssumeRole允许假设Cross-Account-Role-B。
   a. 在IAM控制台中 AWS 帐户AccountA，选择Job-Execution-Role-A。
   b. 添加以下 Job-Execution-Role-A 策略语句以便对 Cross-Account-Role-B 角色执行 AssumeRole 操作。

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

```
    ]
}
```

假设角色示例

您可以使用单个代入角色访问账户中的所有 S3 资源，或者在 Amazon EMR 6.11 及更高版本中，您可以配置多个IAM角色，以便在访问不同的跨账户 S3 存储桶时代入。

主题

- 使用一个代入的角色访问 S3 资源
- 使用多个代入角色访问 S3 资源

使用一个代入的角色访问 S3 资源

> **Note**
>
> 当您将任务配置为使用单个代入角色时，整个任务中的所有 S3 资源都将使用该角色，包括`entryPoint`脚本。

如果您想使用单个代入角色访问账户 B 中的所有 S3 资源，请指定以下配置：

1. `fs.s3.customAWSCredentialsProvider`将EMRFS配置指定为`spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRole`
2. 对于 Spark，使用`spark.emr-serverless.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN`和`spark.executorEnv.ASSUME`指定驱动程序和执行程序上的环境变量。
3. 对于 Hive，使用`hive.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN`、`tez.am.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN`、和`tez.task.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN`指定 Hive 驱动程序、Tez 应用程序主服务器和 Tez 任务容器上的环境变量。

以下示例说明如何使用代入角色启动具有跨账户访问权限的EMR无服务器作业。

Spark

以下示例说明如何使用代入的角色启动具有对 S3 的跨账户访问权限的 EMR Serverless Spark 作业。

```
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

以下示例说明如何使用代入的角色启动具有跨账户访问权限的 S EMR erverless Hive 作业。

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "hive": {
            "query": "query_location",
            "parameters": "hive_parameters"
        }
```

```
    }' \
    --configuration-overrides '{
        "applicationConfiguration": [{
            "classification": "hive-site",
            "properties": {
                "fs.s3.customAWSCredentialsProvider":
 "com.amazonaws.emr.serverless.credentialsprovider.AssumeRoleAWSCredentialsProvider",
                "hive.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B",
                "tez.am.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B",
                "tez.task.emr-
serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B"
            }
        }]
    }'
```

使用多个代入角色访问 S3 资源

在 S EMR erverless 版本 6.11.0 及更高版本中，您可以配置多个IAM角色，以便在访问不同的跨账户存储桶时代入。如果您想在账户 B 中使用不同的代入角色访问不同的 S3 资源，请在开始运行任务时使用以下配置：

1. `fs.s3.customAWSCredentialsProvider`将EMRFS配置指定为`com.amazonaws.emr.serverless.credentialsprovider.BucketLevelAssumeRoleCreden`
2. 指定EMRFS配置`fs.s3.bucketLevelAssumeRoleMapping`以定义从 S3 存储桶名称到账户 B 中要担任的IAM角色的映射。该值的格式应为`bucket1->role1;bucket2->role2`。

例如，您可以使用`arn:aws:iam::AccountB:role/Cross-Account-Role-B-1`访问存储桶`bucket1`，使用`arn:aws:iam::AccountB:role/Cross-Account-Role-B-2`访问存储桶`bucket2`。以下示例说明如何通过多个代入角色启动具有跨账户访问权限的EMR无服务器作业。

Spark

以下示例说明如何使用多个代入的角色来创建EMR无服务器 Spark 作业运行。

```
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

以下示例说明如何使用多个假设角色来创建EMR无服务器 Hive 作业运行。

```
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"
```

```
            }
        }]
    }'
```

对EMR无服务器中的错误进行故障排除

使用以下信息来帮助诊断和修复您在使用 Amazon EMR Serverless 时可能遇到的常见问题。

主题

- 错误：超过了最大允许容量的限制。
- 错误：已超过配置的最大容量。请稍后重试。
- 错误：S3 访问被拒绝。请检查作业运行时角色对所需 S3 资源的 S3 访问权限。
- 错误: ModuleNotFoundError: 未命名模块<module>。请参阅用户指南，了解如何在EMR无服务器中使用 python 库。
- 错误：无法担任执行角色<role name>，因为该角色不存在或未设置所需的信任关系。

错误：超过了最大允许容量的限制。

此错误表明 EMR Serverless 无法提交作业，因为应用程序已超过您配置的最大容量限制。提高应用程序的最大容量限制。

错误：已超过配置的最大容量。请稍后重试。

此错误表明 EMR Serverless 无法启动新作业，因为该应用程序已超过您配置的最大容量限制。提高应用程序的最大容量限制。

错误：S3 访问被拒绝。请检查作业运行时角色对所需 S3 资源的 S3 访问权限。

此错误表示您的任务无权访问您的 S3 资源。验证任务运行时角色是否有权访问任务需要使用的 S3 资源。要了解有关运行时角色的更多信息，请参阅Amazon EMR Serverless 的 Job 运行时角色。

错误: ModuleNotFoundError: 未命名模块<module>。请参阅用户指南，了解如何在EMR无服务器中使用 python 库。

此错误表明 Python 模块不可用于 Spark 作业。检查依赖的 Python 库是否可用于该作业。有关如何打包 Python 库的更多信息，请参阅将 Python 库与EMR无服务器一起使用。

错误：无法担任执行角色<role name>，因为该角色不存在或未设置所需的信任关系。

此错误表示您为作业指定的作业运行时角色不存在，或者该角色对EMR无服务器权限没有信任关系。要验证该IAM角色是否存在并验证您是否已正确设置该角色的信任策略，请参阅中的说明Amazon EMR Serverless 的 Job 运行时角色。

通过 EMR Studio 使用EMR无服务器运行交互式工作负载

概述

交互式应用程序是启用了交互式功能的EMR无服务器应用程序。借助 Amazon EMR Serverless 交互式应用程序，您可以使用在 Amazon Studio 中管理的 Jupyter 笔记本电脑执行交互式工作负载。EMR这可以帮助数据工程师、数据科学家和数据分析师使用 EMR Studio 对 Amazon S3 和 Amazon DynamoDB 等数据存储中的数据集进行交互式分析。

EMRServerless 中交互式应用程序的用例包括以下内容：

- 数据工程师使用 EMR Studio 中的IDE经验来创建ETL脚本。该脚本从本地提取数据，转换数据以供分析，并将数据存储在 Amazon S3 中。
- 数据科学家使用笔记本来探索数据集并训练机器学习 (ML) 模型以检测数据集中的异常。
- 数据分析师探索数据集并创建生成每日报告的脚本，以更新业务仪表板等应用程序。

先决条件

要在 EMR Serverless 中使用交互式工作负载，必须满足以下要求：

- EMRAmazon EMR 6.14.0 及更高版本支持无服务器交互式应用程序。
- 要访问您的交互式应用程序、执行您提交的工作负载以及从 EMR Studio 运行交互式笔记本，您需要特定的权限和角色。有关更多信息，请参阅 交互式工作负载所需的权限。

交互式工作负载所需的权限

除了访问 EMR Serverless 所需的基本权限外，您还必须为您的IAM身份或角色配置其他权限：

访问您的交互式应用程序

设置 EMR Studio 的用户和工作区权限。有关更多信息，请参阅《亚马逊EMR管理指南》中的 “配置 EMR Studio 用户权限”。

执行您使用EMR无服务器提交的工作负载

设置作业运行时角色。有关更多信息，请参阅 创建作业运行时角色。

从 EMR Studio 运行交互式笔记本

在 Studio 用户的IAM策略中添加以下额外权限：

- **`emr-serverless:AccessInteractiveEndpoints`**-授予访问和连接您指定为的交互式应用程序的权限`Resource`。从 EMR Studio 工作区连接到EMR无服务器应用程序需要此权限。
- **`iam:PassRole`**-授予访问您在附加到应用程序时计划使用的IAM执行角色的权限。从 EMR Studio 工作区连接到EMR无服务器应用程序需要相应的`PassRole`权限。

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

配置交互式应用程序

使用以下高级步骤在 Amazon Studio 中创建具有 Amazon EMR Studio 交互功能的EMR无服务器应用程序 AWS Management Console.

1. 按照中的开始使用 Amazon EMR Serverless步骤创建应用程序。
2. 然后，从 EMR Studio 启动工作区并作为计算选项连接到EMR无服务器应用程序。有关更多信息，请参阅《EMR无服务器入门》文档步骤 2 中的 “交互式工作负载” 选项卡。

将应用程序连接到 Studio Workspace 时，如果应用程序尚未运行，则该应用程序会自动启动触发。您也可以预先启动应用程序，并在将其连接到工作区之前将其准备就绪。

交互式应用程序的注意事项

- EMRAmazon EMR 6.14.0 及更高版本支持无服务器交互式应用程序。
- EMRStudio 是唯一一款与EMR无服务器交互式应用程序集成的客户端。EMR无服务器交互式应用程序不支持以下 EMR Studio 功能：工作区协作、SQL Explorer 和笔记本的编程执行。
- 只有 Spark 引擎支持交互式应用程序。
- 交互式应用程序支持 Python 3 PySpark 和 Spark Scala 内核。
- 您可以在单个交互式应用程序上运行多达 25 个并发笔记本电脑。
- 没有端点或API接口支持带有交互式应用程序的自托管 Jupyter 笔记本电脑。
- 为了获得优化的启动体验，我们建议您为驱动程序和执行程序配置预先初始化的容量，并预先启动应用程序。在预启动应用程序时，要确保它已准备就绪，以便将其连接到 Workspace。

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

- 默认情况下，已`autoStopConfig`为应用程序启用。这会在空闲时间 30 分钟后关闭应用程序。您可以将此配置作为`create-application`或`update-application`请求的一部分进行更改。
- 使用交互式应用程序时，我们建议您配置预先初始化的内核、驱动程序和执行程序容量以运行笔记本电脑。每个 Spark 交互式会话都需要一个内核和一个驱动程序，因此 EMR Serverless 会为每个预初始化的驱动程序维护一个预先初始化的内核工作程序。默认情况下，即使您没有为驱动程序指定任何预先初始化的容量，EMRServerless 也会在整个应用程序中保持一个内核工作程序的预初始化容量。每个内核工作程序使用 4 v CPU 和 16 GB 的内存。有关当前定价信息，请参阅 Amazon EMR 定价页面。
- 你必须有足够的 v CPU 服务配额 AWS 账户 运行交互式工作负载。如果您不运行支持 Lake Formation 的工作负载，我们建议至少使用 24 v。CPU如果你这样做，我们建议至少使用 28 v CPU。
- EMR如果笔记本电脑的内核闲置时间超过 60 分钟，Serverless 会自动从笔记本电脑中终止内核。EMRServerless 计算笔记本会话期间上次完成的活动后的内核空闲时间。您目前无法修改内核空闲超时设置。
- 要在交互式工作负载中启用 Lake For `spark.emr-serverless.lakeformation.enabled` mation，请在创建EMR无服务器应用程序时将配置设置为`runtime-configuration`对象中

的spark-defaults分类true下。要了解有关在EMR无服务器中启用 Lake Formation 的更多信息，请参阅在亚马逊EMR中启用 Lake For mation。

通过 Apache Livy 端点使用EMR无服务器运行交互式工作负载

在 Amazon 6.14.0 及更高EMR版本中，您可以在创建EMR无服务器应用程序的同时创建和启用 Apache Livy 终端节点，并通过自托管笔记本或自定义客户端运行交互式工作负载。Apache Livy 端点具有以下优点：

- 你可以通过 Jupyter 笔记本安全地连接到 Apache Livy 端点，并使用 Apache Livy 的界面管理 Apache Spark 工作负载。REST
- 将 Apache Livy REST API 操作用于使用来自 Apache Spark 工作负载的数据的交互式 Web 应用程序。

先决条件

要将 Apache Livy 端点与EMR无服务器一起使用，必须满足以下要求：

- 完成 Amazon EMR Serverless 入门中的步骤。
- 要通过 Apache Livy 端点运行交互式工作负载，您需要一定的权限和角色。有关更多信息，请参阅交互式工作负载所需的权限。

所需的权限

除了访问EMR无服务器所需的权限外，您还必须向IAM角色添加以下权限才能访问 Apache Livy 端点和运行应用程序：

- emr-serverless:AccessLivyEndpoints— 授予访问和连接您指定为的启用 Livy 的应用程序的权限。Resource您需要此权限才能运行 Apache Livy 端点提供的RESTAPI操作。
- iam:PassRole— 授予在创建 Apache Livy 会话时访问IAM执行角色的权限。EMRServerless 将使用此角色来执行您的工作负载。
- emr-serverless:GetDashboardForJobRun— 授予生成 Spark Live 用户界面和驱动程序日志链接的权限，并允许访问作为 Apache Livy 会话结果一部分的日志。

```
{
```

```
    "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/*"
        }
    ]
}
```

开始使用

1. 要创建支持 Apache Livy 的应用程序，请运行以下命令。

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

2. EMRServerless 创建您的应用程序后，启动该应用程序以使 Apache Livy 端点可用。

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

使用以下命令检查您的应用程序是否处于状态。状态变为后STARTED，您就可以访问 Apache Livy 端点了。

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

3. 使用以下方式URL访问终端节点：

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

终端节点准备就绪后，您可以根据自己的用例提交工作负载。您必须使用SIGv4协议对发往端点的每个请求进行签名，并传入授权标头。您可以使用以下方法来运行工作负载：

- HTTP客户端 — 您必须使用自定义HTTP客户端提交 Apache Livy 端点API操作。
- Sparkmagic 内核 — 你必须在本地运行 sparkmagic 内核并使用 Jupyter 笔记本提交交互式查询。

HTTP客户

要创建 Apache Livy 会话，您必须在请求正文的conf参数emr-serverless.session.executionRoleArn中提交。以下示例是一个POST /sessions请求示例。

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

下表描述了所有可用的 Apache Livy API 操作。

API操作	描述
GET/会话	返回所有活跃的交互式会话的列表。

API操作	描述
POST/会话	通过 spark 或 pyspark 创建新的交互式会话。
GET/sessions/ <*sessionId* >	返回会话信息。
GET/sessions/ <*sessionId* >/州	返回会话状态。
DELETE/sessions/ <*sessionId* >	停止并删除会话。
GET/sessions/ <*sessionId* >/声明	返回会话中的所有语句。
POST/sessions/ <*sessionId* >/声明	在会话中运行语句。
GET/sessions/ <*sessionId* >/声明/ <*statementId* >	返回会话中指定语句的详细信息。
POST/sessions/ <*sessionId* >/声明/ <*statementId* >/取消	取消此会话中的指定语句。

向 Apache Livy 端点发送请求

您也可以直接从客户端向 Apache Livy 端点发送请求。HTTP这样做可以让你在笔记本之外远程运行用例的代码。

在开始向终端节点发送请求之前，请确保已安装以下库：

```
pip3 install botocore awscrt requests
```

以下是用于将HTTP请求直接发送到端点的 Python 脚本示例：

```
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())
```

火花魔法内核

在你安装 sparkmagic 之前，请确保你已经配置好了 AWS 你要在其中安装 sparkmagic 的实例中的凭证

1. 按照安装步骤安装 sparkmagic。请注意，您只需要执行前四个步骤。
2. sparkmagic 内核支持自定义身份验证器，因此您可以将身份验证器与 sparkmagic 内核集成，以便对每个请求进行签名。SIGv4
3. 安装EMR无服务器自定义身份验证器。

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

4. 现在在 sparkmagic 配置 json 文件URL中提供自定义身份验证器和 Apache Livy 端点的路径。使用以下命令打开配置文件。

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

   以下为示例 `config.json` 文件。

```
{
"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. 启动 Jupyter 实验室。它应该使用您在上一步中设置的自定义身份验证。
6. 然后，您可以运行以下 notebook 命令和您的代码开始使用。

```
%%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
```

在内部，每条指令都通过配置的 Apache Livy 端点调用每个 Apache Livy API 操作。URL然后，您可以根据自己的用例编写说明。

注意事项

通过 Apache Livy 端点运行交互式工作负载时，请考虑以下注意事项。

- EMRServerless 使用呼叫者主体维护会话级隔离。创建会话的呼叫者主体是唯一可以访问该会话的人。要进行更精细的隔离，可以在假设凭据时配置源身份。在这种情况下，EMRServerless 会根据呼叫者主体和源身份强制执行会话级隔离。有关源身份的更多信息，请参阅监控和控制以代入角色执行的操作。
- EMR无服务器版本 6.14.0 及更高版本支持 Apache Livy 端点。
- 只有 Apache Spark 引擎支持 Apache Livy 端点。
- Apache Livy 端点支持 Scala Spark 和。 PySpark
- 默认情况下，在您的应用程序中`autoStopConfig`处于启用状态。这意味着应用程序将在闲置 15 分钟后关闭。您可以将此配置作为`create-application`或`update-application`请求的一部分进行更改。
- 您可以在单个支持 Apache Livy 端点的应用程序上运行多达 25 个并发会话。
- 为了获得最佳的启动体验，我们建议您为驱动程序和执行程序配置预先初始化的容量。
- 在连接到 Apache Livy 端点之前，必须手动启动应用程序。
- 你必须有足够的 v CPU 服务配额 AWS 账户 使用 Apache Livy 端点运行交互式工作负载。我们建议至少 24 v CPU。
- Apache Livy 会话的默认超时时间为 1 小时。如果您在一小时内没有运行语句，Apache Livy 就会删除会话并释放驱动程序和执行器。您无法更改此配置。
- 只有活跃的会话才能与 Apache Livy 端点进行交互。会话完成、取消或终止后，您将无法通过 Apache Livy 端点访问该会话。

日记账记录和监控

监控是维护EMR无服务器应用程序和作业的可靠性、可用性和性能的重要组成部分。您应该从EMR无服务器解决方案的所有部分收集监控数据，以便在出现多点故障时可以更轻松地进行调试。

主题

- 存储日志
- 轮换日志
- 加密日志
- 为 Amazon Serverless 配置 Apache Log4j2 属性 EMR
- 监控EMR无服务器
- 通过以下方式实现EMR无服务器自动化 Amazon EventBridge

存储日志

要在 EMR Serverless 上监控您的任务进度并排除作业故障，您可以选择 EMR Serverless 存储和提供应用程序日志的方式。提交任务运行时，您可以将托管存储、Amazon S3 和 Amazon 指定 CloudWatch 为日志选项。

使用 CloudWatch，您可以指定要使用的日志类型和日志位置，也可以接受默认的类型和位置。有关 CloudWatch 日志的更多信息，请参阅the section called “Amazon CloudWatch”。对于托管存储和 S3 日志记录，下表显示了如果您选择托管存储、Amazon S3 存储桶或两者兼而有之，则可以预期的日志位置和用户界面可用性。

选项	事件日志	容器日志	应用程序用户界面
托管存储	存储在托管存储中	存储在托管存储中	支持
托管存储和 S3 存储桶	存放在两个地方	存储在 S3 存储桶中	支持
Amazon S3 存储桶	存储在 S3 存储桶中	存储在 S3 存储桶中	不支持 [1]

[1] 我们建议您保持 “托管存储” 选项处于选中状态。否则，您将无法使用内置应用程序UIs。

使用托管EMR存储进行无服务器登录

默认情况下，EMRServerless 将应用程序日志安全地存储在亚马逊EMR托管存储中，最长可保存 30 天。

> **Note**
>
> 如果您关闭默认选项，Amazon 将EMR无法代表您对任务进行故障排除。

要从 EMR Studio 中关闭此选项，请取消选择 “允许” AWS “提交作业” 页面的 “其他设置” 部分中的 “将日志保留 30 天” 复选框。

要从中关闭此选项 AWS CLI，请在提交作业运行时使用该`managedPersistenceMonitoringConfiguration`配置。

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

使用 Amazon S3 存储桶进行EMR无服务器登录

在您的任务可以向 Amazon S3 发送日志数据之前，您必须在任务运行时角色的权限策略中包含以下权限。将 *DOC-EXAMPLE-BUCKET-LOGGING* 替换为日志记录存储桶的名称。

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

```
        }
    ]
}
```

要设置 Amazon S3 存储桶来存储来自的日志 AWS CLI，请在开始运行作业时使用该`s3MonitoringConfiguration`配置。为此，请在配置`--configuration-overrides`中提供以下内容。

```
{
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET-LOGGING/logs/"
        }
    }
}
```

对于未启用重试功能的批处理作业，EMRServerless 会将日志发送到以下路径：

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

EMRServerless 版本 7.1.0 及更高版本支持流式作业和批处理作业的重试尝试。如果您在启用重试的情况下运行作业，EMRServerless 会自动在日志路径前缀中添加尝试次数，以便您可以更好地区分和跟踪日志。

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

使用 Amazon 进行EMR无服务器登录 CloudWatch

当您向EMR无服务器应用程序提交任务时，可以选择 Amazon CloudWatch 作为存储应用程序日志的选项。这允许您使用 CloudWatch 日志分析功能，例如 Logs Insights 和 Live Tail。 CloudWatch 您也可以将日志从其他系统流式传输 CloudWatch 到其他系统，例如 OpenSearch 进行进一步分析。

EMRServerless 为驱动程序日志提供实时日志记录。您可以使用 live tail 功能或通过 tai CloudWatch CLI l 命令 CloudWatch 实时查看日志。

默认情况下，EMR无服务器的 CloudWatch 日志记录处于禁用状态。要启用它，请参阅中的配置AWS CLI。

> **Note**
>
> Amazon 实时 CloudWatch 发布日志，因此它会从工作人员那里获得更多资源。如果您选择较低的工作人员容量，则对作业运行时间的影响可能会增加。如果您启用 CloudWatch 日志记录，我们建议您选择更大的工作人员容量。如果每秒交易量 (TPS) 速率太低，日志发布也可能会受到限制。PutLogEvents所有服务（包括EMR无服务器）的 CloudWatch 限制配置都是全局的。有关更多信息，请参阅如何确定日志中的限制？ CloudWatch on AWS re: post 。

使用登录所需的权限 CloudWatch

在您的任务可以向 Amazon 发送日志数据之前 CloudWatch，您必须在任务运行时角色的权限策略中包含以下权限。

```
{
    "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

要将 Amazon 设置为存储 CloudWatch 来自 EMR Serverless 的日志 AWS CLI，请在开始运行作业时使用该`cloudWatchLoggingConfiguration`配置。为此，请提供以下配置替代项。或者，您还可以提供日志组名称、日志流前缀名称、日志类型和加密密钥ARN。

如果您未指定可选值，则使用默认日志流将日志 CloudWatch 发布到默认日志组`/aws/emr-serverless/applications/`*`applicationId`*`/jobs/`*`jobId`*`/`*`worker-type`*。

EMRServerless 版本 7.1.0 及更高版本支持流式作业和批处理作业的重试尝试。如果您为作业启用了重试功能，EMRServerless 会自动在日志路径前缀中添加尝试次数，以便您可以更好地区分和跟踪日志。

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

以下显示了使用EMR无服务器的默认设置开启 Amazon CloudWatch 日志记录所需的最低配置：

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

以下示例显示了您在开启 EMR Serverless 的 Amazon CloudWatch 日志记录时可以指定的所有必需和可选配置。此示例下方还列出了支持的`logTypes`值。

```
{
    "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
             }
         }
     }
```

```
}
```

默认情况下，EMRServerless 仅向发布驱动程序 stdout 和 stderr 日志。 CloudWatch如果需要其他日志，则可以在该`logTypes`字段中指定容器角色和相应的日志类型。

以下列表显示了您可以为`logTypes`配置指定的支持的工作器类型：

Spark

- `SPARK_DRIVER : ["STDERR", "STDOUT"]`
- `SPARK_EXECUTOR : ["STDERR", "STDOUT"]`

Hive

- `HIVE_DRIVER : ["STDERR", "STDOUT", "HIVE_LOG", "TEZ_AM"]`
- `TEZ_TASK : ["STDERR", "STDOUT", "SYSTEM_LOGS"]`

轮换日志

Amazon EMR Serverless 可以轮换 Spark 应用程序日志和事件日志。日志轮换有助于解决长时间运行的作业生成可能占用所有磁盘空间的大型日志文件的问题。轮换日志可以帮助您节省磁盘存储空间并减少由于磁盘上没有剩余空间而导致的任务失败次数。

默认情况下，日志轮换处于启用状态，并且仅适用于 Spark 作业。

Spark 事件日志

> **Note**
>
> Spark 事件日志轮换适用于所有 Amazon EMR 版本标签。

EMRServerless 不会生成单个事件日志文件，而是按固定的时间间隔轮换事件日志，并删除较旧的事件日志文件。轮换日志不会影响上传到 S3 存储桶的日志。

Spark 应用程序日志

> **Note**
>
> Spark 应用程序日志轮换适用于所有 Amazon EMR 版本标签。

EMRServerless 还会轮换驱动程序和执行程序（例如stdout和文件）的 Spark 应用程序日志。stderr您可以通过使用 Spark History Server 和 Live UI 链接在 Studio 中选择日志链接来访问最新的日志文件。日志文件是最新日志的截断版本。要查看较旧的轮换日志，您必须在存储日志时指定 Amazon S3 位置。有关更多信息，请参阅使用 Amazon S3 存储桶进行EMR无服务器日志记录。

您可以在以下位置找到最新的日志文件。EMR无服务器每 15 秒刷新一次文件。这些文件的范围可以从 0 MB 到 128 MB 不等。

```
<example-S3-logUri>/applications/<application-id>/jobs/<job-id>/SPARK_DRIVER/stderr.gz
```

以下位置包含较旧的旋转文件。每个文件的大小为 128 MB。

```
<example-S3-logUri>/applications/<application-id>/jobs/<job-id>/SPARK_DRIVER/archived/
stderr_<index>.gz
```

同样的行为也适用于 Spark 执行器。此更改仅适用于 S3 日志记录。日志轮换不会对上传到 Amazon 的日志流进行任何更改 CloudWatch。

EMRServerless 版本 7.1.0 及更高版本支持流式处理和批处理作业的重试尝试。如果您对作业启用了重试功能，EMRServerless 会在此类作业的日志路径中添加前缀，以便您可以更好地跟踪和区分日志。此路径包含所有轮换的日志。

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

加密日志

使用托管EMR存储加密无服务器日志

要使用自己的密KMS钥加密托管存储中的日志，请在提交作业运行时使用managedPersistenceMonitoringConfiguration配置。

```
{
    "monitoringConfiguration": {
        "managedPersistenceMonitoringConfiguration" : {
            "encryptionKeyArn": "key-arn"
        }
    }
}
```

使用 Amazon S3 存储桶加密EMR无服务器日志

要使用自己的密KMS钥加密 Amazon S3 存储桶中的日志，请在提交任务运行时使用`s3MonitoringConfiguration`配置。

```
{
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET-LOGGING/logs/",
            "encryptionKeyArn": "key-arn"
        }
    }
}
```

使用 Amazon 加密EMR无服务器日志 CloudWatch

要使用自己的密KMS钥加密 CloudWatch Amazon 中的日志，请在提交任务运行时使用`cloudWatchLoggingConfiguration`配置。

```
{
    "monitoringConfiguration": {
        "cloudWatchLoggingConfiguration": {
            "enabled": true,
            "encryptionKeyArn": "key-arn"
         }
     }
}
```

日志加密所需的权限

本节内容

- 所需的用户权限
- Amazon S3 和托管存储的加密密钥权限
- Amazon 的加密密钥权限 CloudWatch

所需的用户权限

提交作业、查看日志或应用程序的用户UIs必须具有使用密钥的权限。您可以在KMS密钥策略或策略中为用户、组或角色指定权限。IAM如果提交作业的用户缺少KMS密钥权限，则 EMR Serverless 会拒绝提交作业运行。

密钥策略示例

以下密钥策略提供对kms:GenerateDataKey和的权限kms:Decrypt：

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

IAM策略示例

以下IAM策略提供对kms:GenerateDataKey和的权限kms:Decrypt：

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

要启动 Spark 或 Tez 用户界面，必须向用户、群组或角色授予访问权限，emr-serverless:GetDashboardForJobRunAPI如下所示：

```
{
```

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

Amazon S3 和托管存储的加密密钥权限

在托管存储或 S3 存储桶中使用自己的加密密钥加密日志时，必须按以下方式配置KMS密钥权限。

`emr-serverless.amazonaws.com`委托人必须在KMS密钥的策略中拥有以下权限：

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

作为安全最佳实践，我们建议您在密钥策略中添加`aws:SourceArn`条件KMS密钥。IAM全局条件密钥`aws:SourceArn`有助于确保 EMR Serverless 仅将KMS密钥用于应用程序ARN。

作业运行时角色的IAM策略中必须具有以下权限：

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

```
     "Action": [
           "kms:GenerateDataKey",
           "kms:Decrypt"
      ],
     "Resource": "key-arn"
   }
}
```

Amazon 的加密密钥权限 CloudWatch

要将KMS密钥关联ARN到您的日志组，请对作业运行时角色使用以下IAM策略。

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

配置KMS密钥策略以向 Amazon 授予KMS权限 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:*"
            }
        }
    }
}
```

为 Amazon Serverless 配置 Apache Log4j2 属性 EMR

本页介绍如何为无服务器作业配置自定义 Apache Log4j 2.x 属性，网址为。EMR StartJobRun如果要在应用程序级别配置 Log4j 分类，请参阅。EMR无服务器的默认应用程序配置

为 Amazon Serverless 配置 Spark Log4j2 属性 EMR

在亚马逊EMR版本 6.8.0 及更高版本中，您可以自定义 Apache Log4j 2.x 属性以指定精细的日志配置。这简化了EMR无服务器上的 Spark 作业的故障排除。要配置这些属性，请使用spark-driver-log4j2和spark-executor-log4j2分类。

主题

- Spark 的 log4j2 分类
- Spark 的 Log4j2 配置示例
- Spark 作业示例中的 log4j2
- Log4j2 Spark 的注意事项

Spark 的 log4j2 分类

要自定义 Spark 日志配置，请使用以下分类applicationConfiguration。要配置 Log4j 2.x 属性，请使用以下命令。properties

spark-driver-log4j2

这种分类在log4j2.properties文件中为驱动程序设置值。

spark-executor-log4j2

这种分类为执行者设置log4j2.properties文件中的值。

Spark 的 Log4j2 配置示例

以下示例说明如何使用提交 Spark 作业`applicationConfiguration`以自定义 Spark 驱动程序和执行器的 Log4j2 配置。

要在应用程序级别而不是在提交作业时配置 Log4j 分类，请参阅。EMR无服务器的默认应用程序配置

```
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 作业示例中的 log4j2

以下代码示例演示了如何在初始化 Spark 应用程序的自定义 Log4j2 配置时创建 Spark 应用程序。

Python

Example -使用 Log4j2 进行 Python 的 Spark 作业

```
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()
```

要在执行 Spark 作业时为驱动程序自定义 Log4j2，可以使用以下配置：

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

```
}
```

Scala

Example -使用 Log4j2 进行 Scala 的 Spark 作业

```
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()
  }
}
```

要在执行 Spark 作业时为驱动程序自定义 Log4j2，可以使用以下配置：

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

Log4j2 Spark 的注意事项

以下 Log4j2.x 属性不可针对 Spark 进程进行配置：

- `rootLogger.appenderRef.stdout.ref`

- appender.console.type
- appender.console.name
- appender.console.target
- appender.console.layout.type
- appender.console.layout.pattern

有关您可以配置的 Log4j2.x 属性的详细信息，请参阅上的文件。log4j2.properties.template GitHub

监控EMR无服务器

本节介绍监控您的 Amazon EMR 无服务器应用程序和任务的方法。

主题

- 监控EMR无服务器应用程序和作业
- 使用适用于 Prometheus 的亚马逊托管服务监控 Spark 指标
- EMR无服务器使用率指标

监控EMR无服务器应用程序和作业

借助适用于EMR无服务器的 Amazon CloudWatch 指标，您可以接收 1 分钟的 CloudWatch 指标并访问 CloudWatch 控制面板，以查看您的EMR无服务器应用程序的 near-real-time操作和性能。

EMR无服务器向 CloudWatch 每分钟发送一次指标。EMRServerless 在应用程序级别以及作业、工作人员类型和级别发布这些指标。 capacity-allocation-type

要开始使用 Serverless GitHub 存储库中提供的EMREMR无服务器 CloudWatch 仪表板模板并进行部署。

> **Note**
>
> EMR无服务器交互式工作负载仅启用应用程序级监控，并具有新的工作器类型维度。Spark_Kernel要监控和调试交互式工作负载，您可以在 EMRStudio 工作区中查看日志和 Apache Spark 用户界面。

下表描述了AWS/EMRServerless命名空间中可用的EMR无服务器维度。

EMR无服务器指标的维度

维度	描述	
ApplicationId	筛选EMR无服务器应用程序的所有指标。	
JobId	筛选EMR无服务器作业运行的所有指标。	
WorkerType	筛选给定工作人员类型的所有指标。例如，您可以筛选 Spark 作业SPARK_DRIVER 和筛SPARK_EXECUTORS 选 Spark 作业。	
CapacityAllocationType	筛选给定容量分配类型的所有指标。例如，您可以筛选预先初始化的容量和其他所有OnDemandCapacity 容量。PreInitCapacity	

应用程序级监控

您可以使用 Amazon CloudWatch 指标监控EMR无服务器应用程序级别的容量使用情况。您还可以设置单一视图，以便在 CloudWatch 仪表板中监控应用程序容量使用情况。

EMR无服务器应用程序指标

指标	描述	主要维度	次要维度
CPUAllocated	vCPUs 分配的总数。	ApplicationId	ApplicationId，WorkerType，CapacityAllocationType
IdleWorkerCount	闲置的员工总数。	ApplicationId	ApplicationId，WorkerTyp

指标	描述	主要维度	次要维度
			e ，CapacityA llocationType
MaxCPUAllowed	应用程序CPU允许的最大值。	ApplicationId	不适用
MaxMemory Allowed	应用程序允许的最大内存（以 GB 为单位）。	ApplicationId	不适用
MaxStorag eAllowed	应用程序允许的最大存储空间（以 GB 为单位）。	ApplicationId	不适用
MemoryAllocated	分配的总内存（以 GB 为单位）。	ApplicationId	Applicati onId ，WorkerTyp e ，CapacityA llocationType
PendingCr eationWor kerCount	待创建的工作人员总数。	ApplicationId	Applicati onId ，WorkerTyp e ，CapacityA llocationType
RunningWo rkerCount	应用程序正在使用的员工总数。	ApplicationId	Applicati onId ，WorkerTyp e ，CapacityA llocationType
StorageAl located	分配的总磁盘存储空间（以 GB 为单位）。	ApplicationId	Applicati onId ，WorkerTyp e ，CapacityA llocationType

指标	描述	主要维度	次要维度
TotalWorkerCount	可用的员工总数。	ApplicationId	ApplicationId ，WorkerType ，CapacityAllocationType

作业级监控

Amazon EMR Serverless 将以下作业级别的指标发送到 Amazon CloudWatch 每隔一分钟。您可以按任务运行状态查看聚合作业运行的指标值。每个指标的单位是计数。

EMR无服务器作业级别指标

指标	描述	主要维度
SubmittedJobs	处于 “已提交” 状态的作业数量。	ApplicationId
PendingJobs	处于 “待定” 状态的任务数。	ApplicationId
ScheduledJobs	处于 “已计划” 状态的作业数量。	ApplicationId
RunningJobs	处于 “正在运行” 状态的作业数量。	ApplicationId
SuccessJobs	处于 “成功” 状态的工作数量。	ApplicationId
FailedJobs	处于 “失败” 状态的作业数量。	ApplicationId
CancellingJobs	处于 “正在取消” 状态的任务数。	ApplicationId
CancelledJobs	处于 “已取消” 状态的作业数量。	ApplicationId

您可以使用特定于引擎的应用程序监控正在运行和已完成的EMR无服务器作业的引擎特定指标。UIs当您查看正在运行的作业的用户界面时，您会看到带有实时更新的实时应用程序用户界面。当您查看已完成任务的用户界面时，您会看到永久性应用程序界面。

运行作业

对于正在运行的EMR无服务器作业，您可以查看提供引擎特定指标的实时界面。你可以使用 Apache Spark 用户界面或 Hive Tez 用户界面来监控和调试你的作业。要访问这些内容UIs，请使用 EMR Studio 控制台或通过以下方式请求安全URL终端节点 AWS Command Line Interface.

已完成的作业

对于已完成的EMR无服务器作业，您可以使用 Spark History Server 或 Persitent Hive Tez 用户界面来查看 Spark 或 Hive 作业运行的作业详细信息、阶段、任务和指标。要访问这些内容UIs，请使用 EMR Studio 控制台，或通过请求安全URL终端节点 AWS Command Line Interface.

Job Worker 级别的监控

Amazon EMR Serverless 将AWS/EMRServerless命名空间和指标组中可用的以下作业工作人员级别的Job Worker Metrics指标发送给亚马逊 CloudWatch。EMRServerless 在作业级别、工作人员类型和级别上从单个工作人员那里收集数据点。 capacity-allocation-type 您可以使用ApplicationId作为维度来监控属于同一应用程序的多个作业。

EMR无服务器工作者级别的指标

指标	描述	单位	主要维度	次要维度
WorkerCpuAllocated	在作业运行中为工作人员分配的 v CPU 核心总数。	无	JobId	ApplicationId 、WorkerType 和 CapacityAllocationType
WorkerCpuUsed	工作者在作业运行中使用的 v CPU 内核总数。	无	JobId	ApplicationId 、WorkerType 和 CapacityAllocationType

指标	描述	单位	主要维度	次要维度
WorkerMemoryAllocated	在作业运行中为工作人员分配的总内存（以 GB 为单位）。	千兆字节 (GB)	JobId	ApplicationId 、WorkerType 和 CapacityAllocationType
WorkerMemoryUsed	工作人员在作业运行中使用的总内存（以 GB 为单位）。	千兆字节 (GB)	JobId	ApplicationId 、WorkerType 和 CapacityAllocationType
WorkerEphemeralStorageAllocated	在作业运行中为工作人员分配的临时存储字节数。	千兆字节 (GB)	JobId	ApplicationId 、WorkerType 和 CapacityAllocationType
WorkerEphemeralStorageUsed	工作人员在作业运行中使用的临时存储字节数。	千兆字节 (GB)	JobId	ApplicationId 、WorkerType 和 CapacityAllocationType
WorkerStorageReadBytes	作业运行中工作人员从存储中读取的字节数。	字节	JobId	ApplicationId 、WorkerType 和 CapacityAllocationType

指标	描述	单位	主要维度	次要维度
WorkerStorageWriteBytes	作业运行中从工作人员写入存储空间的字节数。	字节	JobId	ApplicationId 、WorkerType 和 CapacityAllocationType

以下步骤描述了如何查看各种类型的指标。

Console

使用控制台访问您的应用程序 UI

1. 按照控制台入门中的说明，在 EMR Studio 上导航到您的EMR无服务器应用程序。
2. 要查看正在运行的作业的特定引擎应用程序UIs和日志，请执行以下操作：
   a. 选择具有RUNNING状态的工作。
   b. 在申请详情页面上选择职位，或导航至您的职位的任务详情页面。
   c. 在 D isplay UI 下拉菜单下，选择 Spark UI 或 Hive Tez 用户界面以导航到适用于您的作业类型的应用程序用户界面。
   d. 要查看 Spark 引擎日志，请导航到 Spark 用户界面中的 “执行器” 选项卡，然后选择驱动程序的 “日志” 链接。要查看 Hive 引擎日志，请DAG在 Hive Tez 用户界面中选择相应内容的日志链接。
3. 要查看已完成作业的特定引擎应用程序UIs和日志，请执行以下操作：
   a. 选择具有SUCCESS状态的工作。
   b. 在申请的申请详情页面上选择该职位，或导航至该职位的职位详情页面。
   c. 在 D isplay UI 下拉菜单下，选择 Spark History Server 或 Per sistent Hive Tez 用户界面以导航到您的作业类型的应用程序用户界面。
   d. 要查看 Spark 引擎日志，请导航到 Spark 用户界面中的 “执行器” 选项卡，然后选择驱动程序的 “日志” 链接。要查看 Hive 引擎日志，请DAG在 Hive Tez 用户界面中选择相应内容的日志链接。

AWS CLI

要使用访问您的应用程序用户界面 AWS CLI

- 要生成一个URL可用来访问正在运行和已完成的作业的应用程序用户界面，请调用GetDashboardForJobRunAPI。

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

您生成的有效期为一小时。URL

使用适用于 Prometheus 的亚马逊托管服务监控 Spark 指标

在亚马逊EMR版本 7.1.0 及更高版本中，您可以将EMR无服务器与适用于 Prometheus 的亚马逊托管服务集成，以收集无服务器任务和应用程序的 Apache Spark 指标。EMR当您使用以下任一方法提交工作或创建应用程序时，即可使用此集成 AWS 控制台API、EMR无服务器或 AWS CLI.

先决条件

在向亚马逊 Prometheus 托管服务提供 Spark 指标之前，必须满足以下先决条件。

- 为 Prometheus 工作空间创建亚马逊托管服务。此工作空间用作摄取端点。记下URL显示的 Endpoint-远程写入URL。您需要在创建EMR无服务器应用程序URL时指定。
- 要授予您的任务访问适用于 Prometheus 的亚马逊托管服务以进行监控，请将以下策略添加到您的任务执行角色中。

```
{
    "Sid": "AccessToPrometheus",
    "Effect": "Allow",
    "Action": ["aps:RemoteWrite"],
    "Resource": "arn:aws:aps:<AWS_REGION>:<AWS_ACCOUNT_ID>:workspace/<WORKSPACE_ID>"
}
```

设置

要再次使用 AWS 控制台用于创建与适用于 Prometheus 的亚马逊托管服务集成的应用程序

1. 要创建应用程序，请参阅 Amazon EMR Serverless 入门。
2. 在创建应用程序时，选择 “使用自定义设置”，然后通过在要配置的字段中指定信息来配置应用程序。
3. 在 “应用程序日志和指标” 下，选择 “向亚马逊 Prometheus 托管服务提供引擎指标”，然后指定您的远程写入。URL
4. 指定所需的任何其他配置设置，然后选择 “创建并启动应用程序”。

使用 AWS CLI 或EMR无服务器 API

你也可以使用 AWS CLI 或者 EMR Serverless API 可在你运行或命令时将你的EMR无服务器应用程序与适用于 Prometheus 的亚马逊 Prometheus 托管服务集成。`create-application` `start-job-run`

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"
        }
    }
}'
```

prometheusMonitoringConfiguration在您的命令中包含表示EMR无服务器必须使用代理运行 Spark 作业，该代理负责收集 Spark 指标并将其写入适用于 Prometheus 的亚马逊托管服务的remoteWriteUrl终端节点。然后，您可以使用适用于 Prometheus 的亚马逊托管服务中的 Spark 指标进行可视化、提醒和分析。

高级配置属性

EMRServerless 使用 Spark 中名为的组件PrometheusServlet来收集 Spark 指标，并将性能数据转换为与亚马逊 Prometheus 托管服务兼容的数据。默认情况下，EMRServerless 会在 Spark 中设置默认值，并在您使用提交作业时解析驱动程序和执行者指标。PrometheusMonitoringConfiguration

下表描述了在提交向亚马逊 Prometheus 托管服务发送指标的 Spark 任务时可以配置的所有属性。

Spark 属性	默认值	描述
spark.metrics.conf.*.sink.prometheusServlet.class	org.apache.spark.metrics.sink。 PrometheusServlet	Spark 用来向 Prometheus 的亚马逊托管服务发送指标的类。要覆盖默认行为，请指定您自己的自定义类。
spark.metrics.conf.*.source.jvm.class	org.apache.spark.metrics.source。 JvmSource	Spark 用于从底层 Java 虚拟机收集和发送关键指标的类。要停止收集JVM指标，请通过将其设置为空字符串来禁用此属性，例如""。要覆盖默认行为，请指定您自己的自定义类。

Spark 属性	默认值	描述
`spark.metrics.conf.driver.sink.prometheusServlet.path`	/metrics/prometheus	亚马逊 Prometheus 托管服务用来从驱动程序收集指标的独特URL之处。要覆盖默认行为，请指定自己的路径。要停止收集驱动程序指标，请通过将其设置为空字符串来禁用此属性，例如""。
`spark.metrics.conf.executor.sink.prometheusServlet.path`	/metrics/executor/prometheus	亚马逊 Prometheus 托管服务用来从执行者那里收集指标的独特URL之处。要覆盖默认行为，请指定自己的路径。要停止收集执行者指标，请将其设置为空字符串（例如""）来禁用此属性。

有关 Spark 指标的更多信息，请参阅 Apache Spark 指标。

注意事项和限制

使用适用于 Prometheus 的亚马逊托管服务EMR从 Serverless 收集指标时，请考虑以下注意事项和限制。

- 仅支持在 Serverless 中使用适用于 Prometheus EMR 的亚马逊托管服务 AWS 区域 Prometheus 的亚马逊托管服务现已正式上市。
- 运行代理以在适用于 Prometheus 的亚马逊托管服务上收集 Spark 指标需要工作人员提供更多资源。如果您选择较小的工作器规模，例如 one v CPU worker，则您的作业运行时间可能会增加。
- 只有亚马逊 7.1.0 及更高版本才支持将 Prometheus 的亚马逊托管服务EMR与 Serverless 配合使用。EMR

EMR无服务器使用率指标

您可以使用 Amazon CloudWatch 使用量指标来了解您的账户使用的资源。使用这些指标在 CloudWatch 图表和仪表板上可视化您的服务使用情况。

EMR无服务器使用量指标对应于 Service Quotas。您可以配置警报，以在用量接近服务限额时向您发出警报。有关更多信息，请参阅服务配额用户指南中的服务配额和亚马逊 CloudWatch警报。

有关EMR无服务器服务配额的更多信息，请参阅的终端节点和配额 EMR Serverless。

EMRServerless 的服务配额使用量指标

EMRServerless 在AWS/Usage命名空间中发布以下服务配额使用指标。

指标	描述
ResourceCount	您的账户上正在运行的指定资源的总数。资源由与指标关联的维度定义。

EMR无服务器服务配额使用量指标的维度

您可以使用以下维度来完善 EMR Serverless 发布的使用量指标。

维度	值	描述
Service	EMR无服务器	的名字 AWS 服务 其中包含资源。
Type	资源	EMRServerless 报告的实体类型。
Resource	v CPU	EMRServerless 正在跟踪的资源类型。
Class	无	EMRServerless 正在跟踪的资源类别。

通过以下方式实现EMR无服务器自动化 Amazon EventBridge

您可以使用 … Amazon EventBridge 让你实现自动化 AWS 服务 并自动响应系统事件，例如应用程序可用性问题或资源更改。 EventBridge 提供近乎实时的系统事件流，这些事件描述了你的变化 AWS 资源的费用。您可以编写简单规则来指示您关注的事件，并指示要在事件匹配规则时执行的自动化操作。使用 EventBridge，您可以自动：

- 调用 AWS Lambda 函数
- 将事件中继到 Amazon Kinesis Data Streams
- 激活 AWS Step Functions 状态机
- 通知亚马逊SNS主题或亚马逊SQS队列

例如，当你 EventBridge 与 EMR Serverless 一起使用时，你可以激活 AWS Lambda ETL任务成功时运行，或者在ETL任务失败时通知 Amazon SNS 主题。

EMR无服务器会发出三种事件：

- 应用程序状态更改事件 — 发出应用程序每一次状态更改的事件。有关应用程序状态的更多信息，请参阅应用程序状态。
- Job 运行状态更改事件 — 发出作业运行的每一次状态更改的事件。有关更多信息，请参阅 任务运行状态。
- 任务运行重试事件 — 每次重试从 Amazon EMR Serverless 版本 7.1.0 及更高版本中运行的任务都会触发的事件。

EMR无服务器事件 EventBridge示例

EMRServerless 报告的事件的`aws.emr-serverless`值为`source`，如以下示例所示。

应用程序状态更改事件

以下示例事件显示了处于该`CREATING`状态的应用程序。

```
{
    "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
        }
    }
}
```

Job 运行状态更改事件

以下示例事件显示了从状态移动到SCHEDULED状态的作业运行。RUNNING

```
{
    "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"
    }
}
```

Job 运行重试事件

以下是作业运行重试事件的示例。

```
{
    "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"
    }
}
```

标记资源

您可以使用标签为每个资源分配自己的元数据，以帮助您管理您的EMR无服务器资源。本节概述了标签功能，并向您展示了如何创建标签。

主题

- 什么是标签？
- 标记您的资源
- 标签限制
- 使用标签处理 AWS CLI 还有 Amazon EMR Serverless API

什么是标签？

标签是您分配给的标签 AWS 资源。每个标签都由键 和值组成，这两个参数都由您定义。标签使您能够对自己的 AWS 按用途、所有者和环境等属性划分的资源。在具有相同类型的许多资源时，可以根据分配给资源的标签快速识别具体的资源。例如，您可以为 Amazon EMR Serverless 应用程序定义一组标签，以帮助您跟踪每个应用程序的所有者和堆栈级别。我们建议为每个资源类型设计一组一致的标签键。

标签不会自动分配至资源。向资源添加标签后，您可以随时修改标签的值或从资源中移除标签。标签对 Amazon EMR Serverless 没有任何语义意义，严格解释为字符串。如果您添加的标签的键与该资源上现有标签的键相同，则新值会覆盖旧值。

如果您使用IAM，则可以控制自己中的哪些用户 AWS 账户有权管理标签。有关基于标签的访问策略示例，请参阅基于标签的访问控制策略。

标记您的资源

您可以标记新的或现有的应用程序和作业运行。如果你使用的是 Amazon EMR 无服务器API， AWS CLI，或者 AWS SDK，您可以使用相关API操作上的`tags`参数将标签应用于新资源。您可以使用`TagResource`API操作将标签应用于现有资源。

您可使用某些创建资源操作，以在创建资源时为其指定标签。在这种情况下，如果在创建资源期间无法应用标签，则无法创建资源。此机制可确保对于您希望在创建时标记的资源，要么使用指定的标签创建，要么完全不创建。如果您在创建资源时添加了标签，则无需在创建资源后运行自定义标记脚本。

下表描述了可以标记的 Amazon EMR 无服务器资源。

资源	支持标签	支持标签传播	支持在创建时添加标签（Amazon EMR Serverless API、 AWS CLI，以及 AWS SDK)	API用于创建（可以在创建过程中添加标签）
应用程序	是	不是。 与应用程序关联的标签不会传播到提交给该应用程序的作业运行中。	是	`CreateApplication`
任务运行	是	否	是	`StartJobRun`

标签限制

以下基本限制适用于标签：

- 每个资源最多可以有 50 个用户创建的标签。
- 对于每个资源，每个标签键都必须是唯一的，每个标签键只能有一个值。
- 最大密钥长度为 UTF -8 中的 128 个 Unicode 字符。
- 最大值长度为 UTF -8 中的 256 个 Unicode 字符。
- 允许的字符包括字母、数字、可用 UTF -8 表示的空格以及以下字符：_.:/= +-@。
- 标签的键不能是空字符串。标签值可以是空字符串，但是不能是 null。
- 标签键和值区分大小写。
- 请勿使用`AWS:`或任何大写或小写组合，例如键或值的前缀。这些仅限于 AWS 使用。

使用标签处理 AWS CLI 还有 Amazon EMR Serverless API

使用以下内容 AWS CLI 命令或 Amazon EMR Serverless API 操作可为您的资源添加、更新、列出和删除标签。

资源	支持标签	支持标签传播
添加或覆盖一个或多个标签	tag-resource	TagResource
列出资源的标签	list-tags-for-resource	ListTagsForResource
删除一个或多个标签	untag-resource	UntagResource

以下示例说明如何使用标记或取消标记资源 AWS CLI.

为现有应用程序添加标签

以下命令对现有应用程序进行标记。

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

取消标记现有应用程序

以下命令从现有应用程序中删除标签。

```
aws emr-serverless untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

列出资源的标签

以下命令列出与现有资源关联的标签。

```
aws emr-serverless list-tags-for-resource --resource-arn resource_ARN
```

EMR无服务器教程

本节介绍使用EMR无服务器应用程序时的常见用例。

主题

- 在亚马逊EMR无服务器中使用 Java 17
- 将 Apache Hudi 与无服务器一起使用 EMR
- 将 Apache Iceberg 与无服务器一起使用 EMR
- 将 Python 库与EMR无服务器一起使用
- 在EMR无服务器中使用不同的 Python 版本
- 在EMR无服务器中使用 Delta L OSS ake
- 通过 Airflow 提交EMR无服务器作业
- 在无服务器中使用 Hive 用户定义的EMR函数
- 在 S EMR erverless 中使用自定义镜像
- 在亚马逊无服务器上使用 Apache Spark 的 Amazon Redshift 集成 EMR
- 使用 Amazon Serverless 连接到 DynamoDB EMR

在亚马逊EMR无服务器中使用 Java 17

在亚马逊EMR版本 6.11.0 及更高版本中，您可以将EMR无服务器 Spark 任务配置为使用 Java 虚拟机的 Java 17 运行时（ ）。JVM使用以下方法之一将 Spark 配置为 Java 17。

`JAVA_HOME`

要覆盖 EMR Serverless 6.11.0 及更高版本的JVM设置，您可以为其`spark.emr-serverless.driverEnv`和`spark.executorEnv`环境分类提供该`JAVA_HOME`设置。

x86_64

设置所需的属性以指定 Java 17 作为 Spark 驱动程序和执行程序的`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

设置所需的属性以指定 Java 17 作为 Spark 驱动程序和执行程序的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，你可以在spark-defaults分类中指定 Java 17 来覆盖EMR无服务器 6.11.0 及更高版本的JVM设置。

x86_64

在spark-defaults分类中指定 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分类中指定 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/"
            }
         }
      ]
}
```

将 Apache Hudi 与无服务器一起使用 EMR

将 Apache Hudi 与无服务器应用程序一起EMR使用

1. 在相应的 Spark 作业运行中设置所需的 Spark 属性。

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

2. 要将 Hudi 表同步到已配置的目录，请指定 AWS Glue Data Catalog 作为您的元数据仓，或者配置外部元数据仓。EMR无服务器支持hms作为 Hudi 工作负载的 Hive 表的同步模式。EMRServerless 会将此属性作为默认值激活。要了解有关如何设置元数据仓的更多信息，请参阅元数据仓配置。

> **Important**
>
> EMRServerless 不支持 Hive 表HIVEQL或JDBC将其作为同步模式选项来处理 Hudi 工作负载。要了解更多信息，请参阅同步模式。

   当你使用 AWS Glue Data Catalog 作为你的元数据库，你可以为 Hudi 作业指定以下配置属性。

```
--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
```

要了解有关亚马逊 Apache Hudi 版本的更多信息EMR，请参阅 Hudi 发布历史记录。

将 Apache Iceberg 与无服务器一起使用 EMR

将 Apache Iceberg 与无服务器应用程序一起EMR使用

1. 在相应的 Spark 作业运行中设置所需的 Spark 属性。

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

2. 指定 AWS Glue Data Catalog 作为您的元数据仓或配置外部元数据仓。要了解有关设置元数据仓的更多信息，请参阅元数据仓配置。

   配置要用于 Iceberg 的元数据仓属性。例如，如果你想使用 AWS Glue 数据目录，在应用程序配置中设置以下属性。

```
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
```

   当你使用 AWS Glue Data Catalog 作为你的元数据库，你可以为 Iceberg 作业指定以下配置属性。

```
--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
```

要了解有关亚马逊 Apache Iceberg 版本的更多信息EMR，请参阅 Iceberg 发布历史记录。

将 Python 库与EMR无服务器一起使用

在 Amazon EMR 无服务器应用程序上运行 PySpark 作业时，您可以将各种 Python 库打包为依赖项。为此，您可以使用原生 Python 功能、构建虚拟环境或直接将 PySpark 作业配置为使用 Python 库。本页涵盖了每种方法。

使用原生 Python 功能

当你设置以下配置时，你可以使用 PySpark 将 Python 文件 (.py)、压缩的 Python 包 (.zip) 和 Egg 文件 (.egg) 上传到 Spark 执行器。

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

有关如何使用 Python 虚拟环境执行 PySpark 作业的更多详细信息，请参阅使用 PySpark 原生功能。

构建 Python 虚拟环境

要为一项 PySpark 作业打包多个 Python 库，可以创建隔离的 Python 虚拟环境。

1. 要构建 Python 虚拟环境，请使用以下命令。所示示例将软件包scipy和安装matplotlib到虚拟环境包中，并将存档复制到 Amazon S3 位置。

> **Important**
>
> 你必须在类似的 Amazon Linux 2 环境中运行以下命令，该环境的 Python 版本与你在 EMR Serverless 中使用的 Python 版本相同，即适用于亚马EMR逊 6.6.0 版本的 Python 3.7.10。您可以在EMR无服务器示例存储库中找到 Dockerfile 示例。 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 作业，并将您的属性设置为使用 Python 虚拟环境。

```
--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
```

请注意，如果您不覆盖原始的 Python 二进制文件，则前面设置序列中的第二个配置将是`--conf spark.executorEnv.PYSPARK_PYTHON=python`。

有关如何使用 Python 虚拟环境进行 PySpark 作业的更多信息，请参阅使用 Virtualenv。有关如何提交 Spark 作业的更多示例，请参阅Spark 职位。

将 PySpark 作业配置为使用 Python 库

在 Amazon 6.12.0 及更高EMR版本中，您可以直接将EMR无服务器 PySpark作业配置为使用流行的数据科学 Python 库，例如 p andas NumPy、 ，而PyArrow无需进行任何其他设置。

以下示例说明如何为 PySpark作业打包每个 Python 库。

NumPy

NumPy 是一个用于科学计算的 Python 库，它为数学、排序、随机模拟和基本统计提供多维数组和运算。要使用 NumPy，请运行以下命令：

```
import numpy
```

pandas

pandas 是一个 Python 库，它建立在. NumPy 熊猫图书馆为数据科学家提供了DataFrame数据结构和数据分析工具。要使用 pandas，请运行以下命令：

```
import pandas
```

PyArrow

PyArrow 是一个 Python 库，用于管理内存中的列式数据以提高工作性能。 PyArrow 基于 Apache Arrow 跨语言开发规范，这是一种以列式格式表示和交换数据的标准方法。要使用 PyArrow，请运行以下命令：

```
import pyarrow
```

在EMR无服务器中使用不同的 Python 版本

除了中的用例外将 Python 库与EMR无服务器一起使用，您还可以使用 Python 虚拟环境来处理不同于 Amazon EMR Serverless 应用程序的亚马逊版本中打包的EMR版本的 Python 版本。为此，你必须使用你想要使用的 Python 版本来构建 Python 虚拟环境。

从 Python 虚拟环境中提交作业

1. 使用以下示例中的命令构建您的虚拟环境。此示例将 Python 3.9.9 安装到虚拟环境包中，并将存档复制到 Amazon S3 的位置。

> **Important**
>
> 如果您使用亚马逊 7.0.0 及更高EMR版本，则必须在类似于EMR无服务器应用程序的 Amazon Linux 2023 环境中运行命令。
> 如果您使用的是 6.15.0 或更低版本，则必须在类似的 Amazon Linux 2 环境中运行以下命令。

```
# 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. 将您的属性设置为使用 Python 虚拟环境并提交 Spark 作业。

```
# 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
```

有关如何使用 Python 虚拟环境进行 PySpark 作业的更多信息，请参阅使用 Virtualenv。有关如何提交 Spark 作业的更多示例，请参阅Spark 职位。

在EMR无服务器中使用 Delta L OSS ake

亚马逊 6.9.0 及更高EMR版本

> **Note**
>
> 亚马逊 EMR 7.0.0 及更高版本使用 Delta Lake 3.0.0，它将文件重命名为。`delta-core.jar` `delta-spark.jar`如果您使用的是 Amazon EMR 7.0.0 或更高版本，请务必`delta-spark.jar`在配置中指定。

Amazon EMR 6.9.0 及更高版本包括 Delta Lake，因此您不再需要自己打包 Delta Lake 或为EMR无服务器任务提供`--packages`旗帜。

1. 提交 EMR Serverless 作业时，请确保您具有以下配置属性，并在`sparkSubmitParameters`字段中包含以下参数。

```
--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. 创建本地`delta_sample.py`以测试创建和读取 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，将`delta_sample.py`文件上传到您的 Amazon S3 存储桶。然后使用`start-job-run`命令向现有的EMR无服务器应用程序提交作业。

```
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"
            }
       }'
```

要在 Delta Lake 中使用 Python `delta-core` 库，您可以通过将其打包为依赖项或将其用作自定义映像来添加库。

或者，您可以使用从`delta-core`JAR文件中`SparkContext.addPyFile`添加 Python 库：

```
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])
```

亚马逊 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，请按照以下步骤在您的EMR无服务器应用程序中使用 Delta L OSS ake。

1. 要在您的亚马逊EMR无服务器应用程序上构建与 Spark 版本兼容的 Delta La ke 开源版本，请导航到 Delta GitHub 并按照说明进行操作。
2. 将三角洲湖库上传到您的 Amazon S3 存储桶 AWS 账户.
3. 在应用程序配置中提交EMR无服务器任务时，请包括存储桶中现在的 Delta Lake JAR 文件。

   ```
   --conf spark.jars=s3://DOC-EXAMPLE-BUCKET/jars/delta-core_2.12-1.1.0.jar
   ```
4. 为确保您可以对 Delta 表进行读取和写入，请运行示例 PySpark测试。

   ```
   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
```

通过 Airflow 提交EMR无服务器作业

Apache Airflow 中的亚马逊提供商提供EMR无服务器运营商。有关运营商的更多信息，请参阅 Apache Airflow 文档中的亚马逊EMR无服务器运营商。

您可以使用`EmrServerlessCreateApplicationOperator`创建 Spark 或 Hive 应用程序。您还可以使用`EmrServerlessStartJobOperator`使用新应用程序启动一项或多项作业。

要在带有 Airflow 2.2 的 Apache Airflow (MWAA) 的亚马逊托管工作流程 () 中使用运算符，请在`requirements.txt`文件中添加以下行并更新您的MWAA环境以使用新文件。

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

请注意，Amazon提供商的5.0.0版本中增加了EMR无服务器支持。6.0.0 版本是与 Airflow 2.2.2 兼容的最新版本。您可以在 Airflow 2.4.3 开启的情况下使用更高版本。MWAA

以下简短示例显示了如何创建应用程序、运行多个 Spark 作业然后停止该应用程序。完整的示例可在EMR无服务器示例 GitHub 存储库中找到。有关`sparkSubmit`配置的更多详细信息，请参阅Spark 职位。

```
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 = "DOC-EXAMPLE-BUCKET"
```

```
DEFAULT_MONITORING_CONFIG = {
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {"logUri": f"s3://DOC-EXAMPLE-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)
```

在无服务器中使用 Hive 用户定义的EMR函数

Hive 用户定义函数 (UDFs) 允许您创建自定义函数来处理记录或记录组。在本教程中，您将使用UDF带有预先存在的 Amazon EMR Serverless 应用程序的示例来运行输出查询结果的作业。要了解如何设置应用程序，请参阅开始使用 Amazon EMR Serverless。

将UDF与EMR无服务器一起使用

1. 导航至GitHub查看示例UDF。克隆存储库并切换到要使用的 git 分支。更新存储库`pom.xml`文件`maven-compiler-plugin`中的以获得源代码。同时将目标 Java 版本配置更新为`1.8`。运行`mvn package -DskipTests`以创建包含您的样本的JAR文件UDFs。
2. 创建JAR文件后，使用以下命令将其上传到您的 S3 存储桶。

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

3. 创建示例文件以使用其中一个示例UDF函数。将此查询另存为`udf_example.q`，并将其上传到您的 S3 存储桶。

   ```
   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 Hive 作业。

   ```
   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命令检查作业的状态。等待状态更改为SUCCESS。

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

6. 使用以下命令下载输出文件。

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

该stdout.gz文件类似于以下内容。

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

在 S EMR erverless 中使用自定义镜像

主题

- 使用自定义 Python 版本
- 使用自定义 Java 版本
- 构建数据科学映像
- 使用 Apache Sedona 处理地理空间数据

使用自定义 Python 版本

您可以构建自定义镜像以使用不同版本的 Python。例如，要将 Python 3.10 版本用于 Spark 作业，请运行以下命令：

```
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
```

在提交 Spark 作业之前，请将您的属性设置为使用 Python 虚拟环境，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
```

使用自定义 Java 版本

以下示例演示如何生成自定义镜像，以便将 Java 11 用于 Spark 作业。

```
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
```

在提交 Spark 作业之前，请将 Spark 属性设置为使用 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-
```

构建数据科学映像

以下示例说明如何包含常见的数据科学 Python 包，例如 Pandas 和 NumPy。

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

USER root

# python packages
RUN pip3 install boto3 pandas numpy
RUN pip3 install -U scikit-learn==0.23.2 scipy
RUN pip3 install sk-dist
RUN pip3 install xgboost

# EMR Serverless will run the image as hadoop
USER hadoop:hadoop
```

使用 Apache Sedona 处理地理空间数据

以下示例说明如何构建包含用于地理空间处理的 Apache Sedona 的图像。

```
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
```

在亚马逊无服务器上使用 Apache Spark 的 Amazon Redshift 集成 EMR

在亚马逊6.9.0及更高EMR版本中，每张发布的图片都包含A pache Spark和Amazon Redshift 之间的连接器。使用此连接器，您可以使用 Amazon EMR Serverless 上的 Spark 来处理存储在 Amazon Redshift 中的数据。集成基于 `spark-redshift` 开源连接器。对于亚马逊 EMR Serverless，适用于 A pache Spark 的 Amazon Redshift 集成作为原生集成包含在内。

主题

- 启动集成了 Apache Spark 的 Amazon Redshift 的 Spark 应用程序
- 使用适用于 Apache Spark 的 Amazon Redshift 集成进行身份验证
- 在 Amazon Redshift 中进行读取和写入
- 使用 Spark 连接器时的注意事项和限制

启动集成了 Apache Spark 的 Amazon Redshift 的 Spark 应用程序

要使用与 EMR Serverless 6.9.0 的集成，你必须在 Spark 作业中传递所需的 Spark-Redshift 依赖关系。用于包含`--jars`与 Redshift 连接器相关的库。要查看 `--jars` 选项支持的其他文件位置，请参阅 Apache Spark 文档的 Advanced Dependency Management（高级依赖项管理）部分。

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

Amazon 6.10.0 及更高EMR版本不需要`minimal-json.jar`依赖关系，并且默认情况下会自动将其他依赖项安装到每个集群。以下示例显示了如何使用适用于 Apache Spark 的 Amazon Redshift 集成启动 Spark 应用程序。

Amazon EMR 6.10.0 +

在EMR无服务器版本 6.10.0 及更高版本上使用 Apache Spark 的 Amazon Redshift 集成，在亚马逊无服务器EMR上启动 Spark 作业。

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

Amazon EMR 6.9.0

要在 EMR Serverless 版本 6.9.0 上通过与 Apache Spark 集成 Apache Spark 的 Amazon Redshift 在亚马逊无服务器EMR上启动 Spark 作业，请使用以下示例`--jars`所示的选项。请注意，`--jars`选项中列出的路径是JAR文件的默认路径。

```
--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
```

使用适用于 Apache Spark 的 Amazon Redshift 集成进行身份验证

使用 AWS Secrets Manager 检索凭证并连接到亚马逊 Redshift

你可以通过将凭证存储在 Secrets Manager 中来安全地向 Amazon Redshift 进行身份验证，然后让 Spark 作业调用获取证书：`GetSecretValue`API

```
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
```

使用驱动程序向 Amazon Redshift 进行身份验证 JDBC

在里面设置用户名和密码 JDBC URL

您可以通过在中指定 Amazon Redshift 数据库名称和密码来向 Amazon Redshift 集群对 Spark 任务进行身份验证。JDBC URL

> **Note**
>
> 如果您在中传递了数据库证书URL，则有权访问的任何人URL也可以访问这些证书。通常不建议使用此方法，因为这不是一个安全的选项。

如果您的应用程序不考虑安全性，则可以使用以下格式在中设置用户名和密码 JDBCURL：

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

对 Amazon EMR 无服务器任务执行角色使用IAM基于身份的身份验证

从亚马逊 EMR Serverless 版本 6.9.0 开始，亚马逊 Redshift JDBC 驱动程序 2.1 或更高版本已打包到环境中。在JDBC驱动程序 2.1 及更高版本中，JDBCURL您可以指定而不包含原始用户名和密码。

相反，您可以指定 `jdbc:redshift:iam://` 方案。这会命令JDBC驱动程序使用您的EMR无服务器任务执行角色自动获取凭证。有关更多信息，请参阅 Amazon Redshift 管理指南中的配置JDBC或ODBC连接以使用IAM证书。这方面的一个例子URL是：

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

满足所提供的条件时，您的任务执行角色需要以下权限：

权限	任务执行角色所需的条件
`redshift:GetClusterCredentials`	JDBC司机需要从亚马逊 Redshift 获取凭证
`redshift:DescribeCluster`	如果您指定了 Amazon Redshift 集群并且，则为必填项 AWS 区域 在JDBCURL而不是端点中
`redshift-serverless:GetCredentials`	JDBC驱动程序需要从亚马逊 Redshift Serverless 获取凭证
`redshift-serverless:GetWorkgroup`	如果您使用的是 Amazon Redshift Serverless，并且要指定工作组名称和URL区域，则为必填项

在另一个地方连接到亚马逊 Redshift VPC

在下VPC设置预配置的 Amazon Redshift 集群或 Amazon Redshift 无服务器工作组时，必须为VPC访问资源的EMR亚马逊无服务器应用程序配置连接。有关如何在EMR无服务器应用程序上配置VPC连接的更多信息，请参阅配置VPC访问权限。

- 如果您预配置的 Amazon Redshift 集群或 Amazon Redshift 无服务器工作组可以公开访问，则可以在创建无服务器应用程序时指定一个或多个已连接网关的私有NAT子网。EMR
- 如果您预配置的 Amazon Redshift 集群或 Amazon Redshift 无服务器工作组无法公开访问，则必须按照中所述为亚马逊 Redshift 集群创建亚马逊 Redshift 托管VPC终端节点。配置VPC访问权限或者，您可以按照《亚马逊 Redshift 管理指南》中连接亚马逊 Redshift Serverless 中所述创建您的亚马逊 Redshift 无服务器工作组。您必须将您的集群或子组与您在创建 S EMR erverless 应用程序时指定的私有子网相关联。

> **Note**
>
> 如果您使用IAM基于基础的身份验证，并且您的EMR无服务器应用程序的私有子网未连接网NAT关，则还必须在这些子网上为 Amazon Redshift 或 Amazon Redshift Serverless 创建VPC终端节点。这样，JDBC驱动程序就可以获取凭证。

在 Amazon Redshift 中进行读取和写入

以下代码示例用于 PySpark 在带有数据源和 Spark 的 Amazon Redshift 数据库中读取API和写入示例数据。SQL

Data source API

PySpark 用于从带有数据源的 Amazon Redshift 数据库中读取和写入示例数据。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

PySpark 用于通过 Spark 读取和写入亚马逊 Redshift 数据库的示例数据。SQL

```
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 连接器时的注意事项和限制

- 我们建议你开启从亚马逊上SSL的 Spark EMR 到亚马逊 Redshift 的JDBC连接。
- 我们建议您在中管理 Amazon Redshift 集群的证书 AWS Secrets Manager 作为最佳实践。参见使用 AWS Secrets Manager 以检索连接到亚马逊 Redshift 的凭证为例。
- 我们建议您传递一个带有 Amazon Redshift 身份验证参数参数参数`aws_iam_role`的IAM角色。
- 参数 `tempformat` 目前不支持 Parquet 格式。
- 它们`tempdir`URI指向亚马逊 S3 的位置。此临时目录不会自动清理，因此可能会增加额外成本。
- 请考虑以下针对 Amazon Redshift 的建议：
  - 建议阻止对 Amazon Redshift 集群的公有访问。

- 建议启用 Amazon Redshift 审计日志记录。
- 建议启用 Amazon Redshift 静态加密。

- 请考虑以下针对 Amazon S3 的建议：
  - 建议阻止对 Amazon S3 存储桶的公有访问。
  - 建议使用 Amazon S3 服务器端加密以加密使用的 Amazon S3 存储桶。
  - 建议使用 Amazon S3 生命周期策略定义 Amazon S3 存储桶的保留规则。
  - Amazon EMR 始终会验证从开源导入镜像的代码。出于安全原因，我们不支持从 Spark 到 Amazon S3 的以下身份验证方法：
    - 设置 AWS `hadoop-env`配置分类中的访问密钥
    - 编码 AWS 中的访问密钥 `tempdir` URI

有关使用连接器及其支持参数的更多信息，请参阅以下资源：

- Amazon Redshift Management Guide（《Amazon Redshift 管理指南》）中的 Amazon Redshift integration for Apache Spark（适用于 Apache Spark 的 Amazon Redshift 集成）
- Github 上的 `spark-redshift` 社区存储库

使用 Amazon Serverless 连接到 DynamoDB EMR

在本教程中，您将来自美国地理名称委员会的数据子集上传到 Amazon S3 存储桶，然后使用 Amazon S EMR erverless 上的 Hive 或 Spark 将数据复制到您可以查询的亚马逊 DynamoDB 表中。

步骤 1：将数据上传到 Amazon S3 存储桶

要创建 Amazon S3 存储桶，请按照《亚马逊简单存储服务控制台用户指南》中创建存储桶中的说明进行操作。将对的*DOC-EXAMPLE-BUCKET*引用替换为您新创建的存储桶的名称。现在，您的EMR无服务器应用程序已准备好运行作业。

1. `features.zip`使用以下命令下载示例数据档案。

   ```
   wget https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/
   features.zip
   ```

2. 从存档中提取`features.txt`文件并查看文件中的前几行：

   ```
   unzip features.zip
   ```

```
head features.txt
```

结果应类似于以下内容。

```
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
```

此处每行中的字段表示唯一标识符、名称、自然特征类型、州、纬度（以度为单位）、经度（以度为单位）和以英尺为单位的高度。

3. 将您的数据上传到亚马逊 S3

```
aws s3 cp features.txt s3://DOC-EXAMPLE-BUCKET/features/
```

步骤 2：创建 Hive 表

使用 Apache Spark 或 Hive 创建一个新的 Hive 表，其中包含在 Amazon S3 中上传的数据。

Spark

要使用 Spark 创建 Hive 表，请运行以下命令。

```
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';")
```

现在，您有一个填充的 Hive 表，其中包含来自该`features.txt`文件的数据。要验证您的数据是否在表中，请运行 Spark SQL 查询，如以下示例所示。

```
sparkSession.sql(
    "SELECT state_alpha, COUNT(*) FROM hive_features GROUP BY state_alpha;")
```

Hive

要使用 Hive 创建 Hive 表，请运行以下命令。

```
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://DOC-EXAMPLE-BUCKET/features';
```

现在，您有一个 Hive 表，其中包含该`features.txt`文件中的数据。要验证您的数据是否在表中，请运行 HiveQL 查询，如以下示例所示。

```
SELECT state_alpha, COUNT(*) FROM hive_features GROUP BY state_alpha;
```

步骤 3：将数据复制到 DynamoDB

使用 Spark 或 Hive 将数据复制到新的 DynamoDB 表中。

Spark

要将您在上一步中创建的 Hive 表中的数据复制到 DynamoDB，请按照将数据复制到 DynamoDB 中的步骤 1-3 进行操作。这将创建一个名为的新 DynamoDB 表。Features然后，您可以直接从文本文件中读取数据并将其复制到您的 DynamoDB 表中，如下例所示。

```
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://DOC-EXAMPLE-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

要将您在上一步中创建的 Hive 表中的数据复制到 DynamoDB，请按照将数据复制到 DynamoDB 中的说明进行操作。

步骤 4：从 DynamoDB 查询数据

使用 Spark 或 Hive 来查询你的 DynamoDB 表。

Spark

要查询您在上一步中创建的 DynamoDB 表中的数据，您可以使用 SQL Spark 或 Spark。MapReduce API

Example — 使用 Spark 查询你的 DynamoDB 表 SQL

以下 Spark SQL 查询按字母顺序返回所有功能类型的列表。

```
val dataFrame = sparkSession.sql("SELECT DISTINCT feature_class \
    FROM ddb_features \
    ORDER BY feature_class;")
```

以下 Spark SQL 查询返回以字母 M 开头的所有湖泊的列表。

```
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;")
```

以下 Spark SQL 查询返回包含至少三个高于一英里的要素的所有状态的列表。

```
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 查询你的 DynamoDB 表 MapReduce API

以下 MapReduce 查询按字母顺序返回所有要素类型的列表。

```
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")
```

以下 MapReduce 查询返回以字母 M 开头的所有湖泊的列表。

```
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")
```

以下 MapReduce 查询返回包含至少三个高于一英里的要素的所有州的列表。

```
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

要查询您在上一步中创建的 DynamoDB 表中的数据，请按照查询 DynamoDB 表中的数据中的说明进行操作。

设置跨账户访问

要为 EMR Serverless 设置跨账户访问权限，请完成以下步骤。在示例中，AccountA是您创建亚马逊EMR无服务器应用程序的账户，AccountB也是您的亚马逊 DynamoDB 所在的账户。

1. 在中创建 DynamoDB 表。AccountB有关更多信息，请参阅步骤 1：创建表。
2. 在中创建一个AccountB可以访问 DynamoDB 表的Cross-Account-Role-BIAM角色。

   a. 登录 AWS Management Console 然后打开IAM控制台，网址为https://console.aws.amazon.com/iam/。

   b. 选择角色，然后创建一个名为的新角色Cross-Account-Role-B。有关如何创建IAM角色的更多信息，请参阅用户指南中的创建IAM角色。

   c. 创建授予访问跨账户 DynamoDB 表权限的IAM策略。然后将该IAM策略附加到Cross-Account-Role-B。

      以下是授予对 DynamoDB 表的访问权限的策略。CrossAccountTable

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

   d. 编辑 Cross-Account-Role-B 角色的信任关系。

      要为角色配置信任关系，请在IAM控制台中为在步骤 2：跨账户角色 B 中创建的角色选择信任关系选项卡。

选择 “编辑信任关系”，然后添加以下策略文档。本文档AccountA允许Job-Execution-Role-A我担任此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Job-Execution-Role-A予AccountA假设- STS Assume role权限Cross-Account-Role-B。

在IAM控制台中 AWS 账户 AccountA，选择Job-Execution-Role-A。添加以下 Job-Execution-Role-A 策略语句以便对 Cross-Account-Role-B 角色执行 AssumeRole 操作。

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

f. 使用核心站点分类com.amazonaws.emr.AssumeRoleAWSCredentialsProvider中的值设置dynamodb.customAWSCredentialsProvider属性。将环境变ASSUME_ROLE_CREDENTIALS_ROLE_ARN量的ARN值设置为Cross-Account-Role-B。

3. 使用Job-Execution-Role-A运行 Spark 或 Hive 作业。

注意事项

将 DynamoDB 连接器与 Apache Spark 配合使用时的注意事项

- Spark SQL 不支持使用存储处理器选项创建 Hive 表。有关更多信息，请参阅 Apache Spark 文档中的指定 Hive 表的存储格式。
- Spark SQL 不支持使用存储处理程序的`STORED BY`操作。如果您想通过外部 Hive 表与 DynamoDB 表进行交互，请先使用 Hive 创建表。
- 要将查询转换为 DynamoDB 查询，DynamoDB 连接器使用谓词下推。谓词下推按映射到 DynamoDB 表分区键的列筛选数据。谓词下推仅在将连接器与 Spark 配合使用时才起SQL作用，而不是。 MapReduce API

在 Apache Hive 中使用 DynamoDB 连接器时的注意事项

调整映射器的最大数量

- 如果您使用`SELECT`查询从映射到 DynamoDB 的外部 Hive 表中读取数据，则 Serverless EMR 上的映射任务数量将计算为为 DynamoDB 表配置的总读取吞吐量除以每个地图任务的吞吐量。每个地图任务的默认吞吐量为 100。
- 根据为 DynamoDB 配置的读取吞吐量，Hive 作业可以使用超出每个EMR无服务器应用程序配置的最大容器数量的地图任务数量。此外，长时间运行的 Hive 查询可能会消耗 DynamoDB 表的所有预配置读取容量。这会对其他用户产生负面影响。
- 您可以使用该`dynamodb.max.map.tasks`属性来设置地图任务的上限。您也可以使用此属性根据任务容器的大小调整每个地图任务读取的数据量。
- 您可以在 Hive 查询级别或start-job-run命令的`hive-site`分类中设置该`dynamodb.max.map.tasks`属性。此值必须等于或大于 1。当 Hive 处理您的查询时，生成的 Hive 作业使用的值不超过它从 DynamoDB 表中读取`dynamodb.max.map.tasks`时的值。

调整每个任务的写入吞吐量

- EMRServerless 上每个任务的写入吞吐量是按为 DynamoDB 表配置的总写入吞吐量除以该属性的值计算得出的。`mapreduce.job.maps`对于 Hive，此属性的默认值为 2。因此，Hive 作业最后阶段的前两个任务可能会消耗所有的写入吞吐量。这会导致限制同一作业或其他作业中其他任务的写入。
- 为避免写入限制，您可以根据最后阶段的任务数或要为每个任务分配的写入吞吐量来设置`mapreduce.job.maps`属性的值。在 S EMR erverless 上的start-job-run命令`mapred-site`分类中设置此属性。

安全性

云安全位于 AWS 是最高优先级。作为 AWS 客户，您将受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方的共同责任 AWS 还有你。责任共担模式将其描述为云的 安全性和云中 的安全性：

- 云安全 — AWS 负责保护运行的基础架构 AWS 中的服务 AWS 云。 AWS 还为您提供可以安全使用的服务。作为安全措施的一部分，第三方审计师会定期测试和验证我们安全的有效性 AWS 合规计划。要了解适用于 Amazon EMR Serverless 的合规计划，请参阅 AWS 按合规计划划分的范围内的服务。
- 云端安全 — 您的责任由以下因素决定 AWS 您使用的服务。您还需要对其它因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

本文档可帮助您了解在使用 Amazon EMR Serverless 时如何应用分担责任模型。本章中的主题向您展示如何配置 Amazon EMR Serverless 以及如何使用其他 AWS 满足您的安全与合规目标的服务。

主题

- Amazon EMR 无服务器的安全最佳实践
- 数据保护
- Amazon EMR Serverless 中的身份和访问管理 (IAM)
- 将EMR无服务器与 AWS Lake Formation 用于精细的访问控制
- 工作者间加密
- 使用EMR无服务器进行数据保护的 Secrets Manager
- 将 Amazon S3 访问权限授予与EMR无服务器配合使用
- 使用记录 Amazon EMR 无服务器API呼叫 AWS CloudTrail
- Amazon EMR Serverless 合规性验证
- Amazon EMR 无服务器中的弹性
- Amazon EMR Serverless 中的基础设施安全
- Amazon EMR Serverless 中的配置和漏洞分析

Amazon EMR 无服务器的安全最佳实践

Amazon EMR Serverless 提供了许多安全功能，供您在制定和实施自己的安全策略时考虑。以下最佳实践是一般指导原则，并不代表完整安全解决方案。这些最佳实践可能不适合环境或不满足环境要求，请将其视为有用的考虑因素而不是惯例。

采用最低权限原则

EMRServerless 为使用IAM角色（例如执行角色）的应用程序提供了精细的访问策略。我们建议向执行角色仅授予任务所需的最低权限集，例如覆盖应用程序和对日志目标的访问权限。我们还建议定期以及在应用程序代码发生变化时审核任务的权限。

隔离不受信任的应用程序代码

EMRServerless 可在属于不同EMR无服务器应用程序的作业之间建立完全的网络隔离。如果需要作业级隔离，可以考虑将作业隔离到不同的EMR无服务器应用程序中。

基于角色的访问控制 (RBAC) 权限

管理员应严格控制EMR无服务器应用程序的基于角色的访问控制 (RBAC) 权限。

数据保护

这些区域有： AWS 责任共担模式适用于 Amazon EMR Serverless 中的数据保护。如本模型所述，AWS 负责保护运行所有内容的全球基础设施 AWS 云。您负责维护对托管在此基础架构上的内容的控制。此内容包括以下各项的安全配置和管理任务 AWS 您使用的服务。有关数据隐私的更多信息，请参阅数据隐私FAQ。有关欧洲数据保护的信息，请参阅 AWS 责任共担模型和GDPR博客文章 AWS 安全博客。

出于数据保护的目的，我们建议您进行保护 AWS 账户凭证并使用设置个人账户 AWS Identity and Access Managemen IAM t ()。这仅向每个用户授予履行其工作职责所需的权限。我们还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证 (MFA)。
- 使用SSL/TLS与之通信 AWS 资源的费用。我们推荐 TLS 1.2 或更高版本。
- 使用API进行设置和用户活动记录 AWS CloudTrail.
- 使用 AWS 加密解决方案，以及其中的所有默认安全控件 AWS 服务的支持。

- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的个人数据。
- 使用 Amazon EMR Serverless 加密选项对静态和传输中的数据进行加密。
- 如果您在访问时需要 FIPS 140-2 经过验证的加密模块 AWS 通过命令行界面或API，使用FIPS端点。有关可用FIPS端点的更多信息，请参阅联邦信息处理标准 (FIPS) 140-2。

我们强烈建议您切勿将敏感的可识别信息（例如您客户的账号）放入自由格式字段（例如名称字段）。这包括当你使用 Amazon EMR Serverless 或其他服务器时 AWS 使用控制台的服务，API， AWS CLI，或 AWS SDKs。您输入到 Amazon EMR Serverless 或其他服务中的任何数据都可能会被提取并包含在诊断日志中。当您URL向外部服务器提供时，请不要在中包含凭据信息URL以验证您对该服务器的请求。

静态加密

数据加密有助于防止未经授权的用户在集群和关联的数据存储系统中读取数据。这包括保存到持久性媒体的数据（称为静态数据）和在网络中传输时可能被拦截的数据（称为传输中的数据）。

数据加密需要密钥和凭证。您可以从多个选项中进行选择，包括由管理的密钥 AWS Key Management Service、由 Amazon S3 管理的密钥以及您提供的自定义提供商提供的密钥和证书。使用时 AWS KMS 作为您的密钥提供商，加密密钥的存储和使用需要付费。有关更多信息，请参阅 AWS KMS 定价。

在指定加密选项前，确定要使用的密钥和凭证管理系统。然后，针对您指定为加密设置一部分的自定义提供程序，来创建密钥和凭证。

对 Amazon S3 中的EMRFS数据进行静态加密

每个EMR无服务器应用程序都使用特定的发行版，其中包括EMRFS（EMR文件系统）。Amazon S3 加密适用于从 Amazon S3 读取和写入EMR的文件系统 (EMRFS) 对象。启用静态加密时，您可以将 Amazon S3 服务器端加密 (SSECSE) 或客户端加密 () 指定为默认加密模式。（可选）您可以使用 Per bucket encryption overrides (每存储桶加密覆盖) 为单个存储桶指定不同的加密方法。无论是否启用了 Amazon S3 加密，传输层安全 (TLS) 都会对EMR集群节点和 Amazon S3 之间传输的EMRFS对象进行加密。如果您使用CSE带有客户管理密钥的 Amazon S3，则用于在EMR无服务器应用程序中运行任务的执行角色必须有权访问该密钥。有关 Amazon S3 加密的深入信息，请参阅《亚马逊简单存储服务开发人员指南》中的使用加密保护数据。

> **Note**
>
> 当你使用时 AWS KMS，则加密密钥的存储和使用将收取费用。有关更多信息，请参阅 AWS KMS 定价。

Amazon S3 服务器端加密

设置 Amazon S3 服务器端加密时，Amazon S3 在向磁盘写入数据时会在对象级别加密数据，并在访问数据时对数据进行解密。有关更多信息SSE，请参阅《Amazon 简单存储服务开发人员指南》中的使用服务器端加密保护数据。

在 Amazon EMR Serverless SSE 中指定时，您可以在两种不同的密钥管理系统之间进行选择：

- SSE-S3- Amazon S3 为您管理密钥。无需在EMR无服务器上进行其他设置。
- SSE-KMS- 你使用 AWS KMS key 来设置适用于EMR无服务器的策略。无需在EMR无服务器上进行其他设置。

要将 AWS KMS 对写入 Amazon S3 的数据进行加密，使用时有两种选择`StartJobRun`API。您可以为写入 Amazon S3 的所有内容启用加密，也可以对写入特定存储桶的数据启用加密。有关更多信息`StartJobRun`API，请参阅EMR无服务器参API考。

要开启 AWS KMS 对您写入 Amazon S3 的所有数据进行加密，请在调用时使用以下命令`StartJobRun`API。

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

要开启 AWS KMS 对写入特定存储桶的数据进行加密，请在调用时使用以下命令`StartJobRun`API。

```
--conf spark.hadoop.fs.s3.bucket.<DOC-EXAMPLE-BUCKET>.enableServerSideEncryption=true
--conf spark.hadoop.fs.s3.bucket.<DOC-EXAMPLE-
BUCKET>.serverSideEncryption.kms.keyId=<kms-id>
```

SSE使用客户提供的密钥 (SSE-C) 不适用于无服务器。EMR

Amazon S3 客户端加密

使用 Amazon S3 客户端加密，Amazon S3 的加密和解密是在每个亚马逊版本中可用的EMRFS客户端中进行的。EMR在对象上载到 Amazon S3 之前对其进行加密，并在下载后对其进行解密。您指定的提供程序会提供客户端使用的加密密钥。客户端可以使用提供的密钥 AWS KMS (CSE-KMS) 或提供客户端根密钥的自定义 Java 类 (CSE-C)。CSE-KMS 和 CSE-C 之间的加密细节略有不同，具体取决于指定的提供程序以及正在解密或加密的对象的元数据。如果您使用CSE带有客户管理密钥的 Amazon S3，则用于在EMR无服务器应用程序中运行任务的执行角色必须有权访问该密钥。可能会KMS收取额外费用。有关这些差异的更多信息，请参阅《Amazon 简单存储服务开发人员指南》中的使用客户端加密保护数据。

本地磁盘加密

使用行业标准的 AES -256 加密算法，使用服务拥有的密钥对存储在临时存储中的数据进行加密。

密钥管理

您可以配置KMS为自动轮换KMS密钥。这会每年转动一次密钥，同时无限期地保存旧密钥，以便您的数据仍然可以被解密。有关更多信息，请参阅转动客户主密钥。

传输中加密

Ama EMR zon Serverless 提供以下特定于应用程序的加密功能：

- Spark
  - 默认情况下，Spark 驱动程序和执行器之间的通信是经过身份验证的，并且是内部的。RPC驱动程序和执行者之间的通信已加密。
- Hive
  - 之间的沟通 AWS Glue 元数据仓和EMR无服务器应用程序通过以下方式发生。TLS

您应该仅允许使用 Amazon S3 存储桶IAM策略上的 aws: SecureTransport 条件通过 HTTPS (TLS) 进行加密连接。

Amazon EMR Serverless 中的身份和访问管理 (IAM)

AWS Identity and Access Management (IAM) 是一个 AWS 服务 可帮助管理员安全地控制对以下内容的访问权限 AWS 资源的费用。IAM管理员控制谁可以通过身份验证（登录）和授权（拥有权限）使用 Amazon EMR Serverless 资源。IAM是一个 AWS 服务 无需支付额外费用即可使用。

主题

- 受众
- 使用身份进行身份验证
- 使用策略管理访问
- EMR无服务器的工作原理 IAM
- 在无服务器中使用服务相关角色 EMR
- Amazon EMR Serverless 的 Job 运行时角色
- EMR无服务器的用户访问策略示例
- 基于标签的访问控制策略
- 无服务器基于身份的策略示例 EMR
- Amazon EMR 无服务器更新至 AWS 托管策略
- 对 Amazon EMR 无服务器身份和访问进行故障排除

受众

你怎么用 AWS Identity and Access Management (IAM) 会有所不同，具体取决于您在 Amazon EMR Serverless 中所做的工作。

服务用户 — 如果您使用 Amazon EMR Serverless 服务完成工作，则您的管理员会为您提供所需的凭证和权限。当您使用更多的 Amazon EMR Serverless 功能来完成工作时，您可能需要额外的权限。了解如何管理访问权限有助于您向管理员请求适合的权限。如果您无法访问 Amazon EMR Serverless 中的某项功能，请参阅对 Amazon EMR 无服务器身份和访问进行故障排除。

服务管理员 — 如果您负责公司的亚马逊EMR无服务器资源，则可能拥有对 Amazon Serv EMR erless 的完全访问权限。您的工作是确定您的服务用户应该访问哪些 Amazon EMR Serverless 功能和资源。然后，您必须向IAM管理员提交更改服务用户权限的请求。查看此页面上的信息以了解的基本概念IAM。要详细了解贵公司如何IAM与 Amazon EMR Serverless 配合使用，请参阅Amazon EMR Serverless 中的身份和访问管理 (IAM)。

IAM管理员 — 如果您是IAM管理员，则可能需要详细了解如何编写策略来管理 Amazon EMR Serverless 的访问权限。要查看可在中使用的 Amazon EMR Serverless 基于身份的策略示例，请参阅。IAM 无服务器基于身份的策略示例 EMR

使用身份进行身份验证

身份验证是您登录的方式 AWS 使用您的身份凭证。您必须经过身份验证（登录到 AWS) 作为 AWS 账户根用户、以IAM用户身份或通过担任IAM角色来完成。

你可以登录 AWS 使用通过身份源提供的凭证作为联合身份。 AWS IAM Identity Center （IAM身份中心）用户、贵公司的单点登录身份验证以及您的 Google 或 Facebook 凭据就是联合身份的示例。当您以联合身份登录时，您的管理员之前使用IAM角色设置了联合身份。当你访问时 AWS 通过使用联合，您就是在间接担任角色。

根据您的用户类型，您可以登录 AWS Management Console 或者 AWS 访问门户。有关登录的更多信息 AWS，请参阅如何登录您的 AWS 账户中的 AWS 登录 用户指南。

如果你访问 AWS 以编程方式， AWS 提供了一个软件开发套件 (SDK) 和一个命令行界面 (CLI)，用于使用您的凭证对您的请求进行加密签名。如果你不使用 AWS 工具，你必须自己签署请求。有关使用推荐的方法自行签署请求的更多信息，请参阅签名 AWS APIIAM用户指南中的请求。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如， AWS 建议您使用多重身份验证 (MFA) 来提高帐户的安全性。要了解更多信息，请参阅中的多重身份验证 AWS IAM Identity Center《用户指南》和《使用多因素身份验证》(MFA) AWS（在 IAM 用户指南中）。

AWS 账户 根用户

当你创建 AWS 账户，您从一个登录身份开始，该身份可以完全访问所有人 AWS 服务 以及账户中的资源。这个身份叫做 AWS 账户 root 用户，使用您创建账户时使用的电子邮件地址和密码登录即可访问。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关需要您以 root 用户身份登录的任务的完整列表，请参阅《用户指南》中的 “需要根用户凭据的IAM任务”。

联合身份

作为最佳实践，要求人类用户（包括需要管理员访问权限的用户）使用与身份提供商的联合身份进行访问 AWS 服务 通过使用临时证书。

联合身份是企业用户目录中的用户、Web 身份提供商、 AWS Directory Service、身份中心目录或任何访问的用户 AWS 服务 通过使用通过身份源提供的凭证。当联合身份访问时 AWS 账户，他们扮演角色，角色提供临时证书。

要进行集中访问管理，我们建议您使用 AWS IAM Identity Center。 您可以在 Ident IAM ity Center 中创建用户和群组，也可以连接并同步到您自己的身份源中的一组用户和群组，以便在您的所有用户和

群组中使用 AWS 账户 和应用程序。有关IAM身份中心的信息，请参阅什么是IAM身份中心？ 在 AWS IAM Identity Center 用户指南。

IAM 用户和组

IAM用户是你内心的身份 AWS 账户 对个人或应用程序具有特定权限。在可能的情况下，我们建议使用临时证书，而不是创建拥有密码和访问密钥等长期凭证的IAM用户。但是，如果您有需要IAM用户长期凭证的特定用例，我们建议您轮换访问密钥。有关更多信息，请参阅《IAM用户指南》中的针对需要长期凭证的用例定期轮换访问密钥。

IAM群组是指定IAM用户集合的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可以拥有一个名为的群组，IAMAdmins并授予该群组管理IAM资源的权限。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅《IAM用户指南》中的何时创建IAM用户（而不是角色）。

IAM角色

IAM角色是你内在的身份 AWS 账户 具有特定权限的。它与IAM用户类似，但与特定人员无关。你可以暂时扮IAM演一个角色 AWS Management Console 通过切换角色。你可以通过调用来扮演角色 AWS CLI 或者 AWS API操作或使用自定义URL。有关使用角色的方法的更多信息，请参阅IAM用户指南中的使用IAM角色。

IAM具有临时证书的角色在以下情况下很有用：

- 联合用户访问 – 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关用于联合身份验证的角色的信息，请参阅IAM用户指南中的为第三方身份提供商创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为了控制您的身份在进行身份验证后可以访问的内容，Ident IAM ity Center 会将权限集关联到中的IAM角色。有关权限集的信息，请参阅中的权限集 AWS IAM Identity Center 用户指南。
- 临时IAM用户权限-IAM 用户或角色可以代入一个IAM角色，为特定任务临时获得不同的权限。
- 跨账户访问-您可以使用IAM角色允许其他账户中的某人（受信任的委托人）访问您账户中的资源。角色是授予跨账户访问权限的主要方式。但是，有些 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解角色和基于资源的跨账户访问策略之间的区别，请参阅IAM用户指南IAM中的跨账户资源访问权限。

- 跨服务访问 — 一些 AWS 服务 使用其他功能 AWS 服务。 例如，当您在服务中拨打电话时，该服务通常会在 Amazon 中运行应用程序EC2或在 Amazon S3 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 转发访问会话 (FAS)-当您使用IAM用户或角色在中执行操作时 AWS，你被视为校长。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS使用委托人的权限调用 AWS 服务，再加上请求的 AWS 服务 向下游服务发出请求。FAS只有当服务收到需要与其他服务进行交互的请求时，才会发出请求 AWS 服务 或需要完成的资源。在这种情况下，您必须具有执行这两个操作的权限。有关提出FAS请求时的政策详情，请参阅转发访问会话。
  - 服务角色-服务IAM角色是服务代替您执行操作的角色。IAM管理员可以在内部创建、修改和删除服务角色IAM。有关更多信息，请参阅创建角色以向某人委派权限 AWS 服务（在 IAM 用户指南中）。
  - 服务相关角色-服务相关角色是一种与服务相关联的服务角色 AWS 服务。 该服务可以代替您执行操作。服务相关角色显示在您的 AWS 账户 并归该服务所有。IAM管理员可以查看但不能编辑服务相关角色的权限。
- 在 Amazon 上运行的应用程序 EC2 — 您可以使用IAM角色管理在EC2实例上运行的应用程序的临时证书 AWS CLI 或者 AWS API请求。这比在EC2实例中存储访问密钥更可取。要分配 AWS 在EC2实例中扮演角色并使其可供其所有应用程序使用，则可以创建附加到该实例的实例配置文件。实例配置文件包含该角色，并允许在EC2实例上运行的程序获得临时证书。有关更多信息，请参阅IAM用户指南中的使用IAM角色向在 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用IAM角色还是使用IAM用户，请参阅《用户指南》中的何时创建IAM角色（而不是IAM用户）。

使用策略管理访问

您可以控制访问权限 AWS 通过创建策略并将其附加到 AWS 身份或资源。策略是中的一个对象 AWS 当与身份或资源关联时，它定义了他们的权限。 AWS 在委托人（用户、root 用户或角色会话）发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都存储在 AWS 作为JSON文件。有关JSON策略文档结构和内容的更多信息，请参阅《IAM用户指南》中的JSON策略概述。

管理员可以使用 AWS JSON用于指定谁有权访问什么的策略。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

默认情况下，用户和角色没有权限。要授予用户对其所需资源执行操作的权限，IAM管理员可以创建IAM策略。然后，管理员可以将IAM策略添加到角色中，用户可以代入这些角色。

IAM无论您使用何种方法执行操作，策略都会定义该操作的权限。例如，假设您有一个允许 `iam:GetRole` 操作的策略。拥有该策略的用户可以从中获取角色信息 AWS Management Console，AWS CLI，或者 AWS API.

基于身份的策略

基于身份的策略是可以附加到身份（例如IAM用户、用户组或角色）的JSON权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅IAM用户指南中的创建IAM策略。

基于身份的策略可以进一步归类为内联策略或托管策略。内联策略直接嵌入单个用户、组或角色中。托管策略是独立的策略，您可以将其附加到您的多个用户、群组和角色 AWS 账户。 托管策略包括 AWS 托管策略和客户托管策略。要了解如何在托管策略或内联策略之间进行选择，请参阅《IAM用户指南》中的在托管策略和内联策略之间进行选择。

基于资源的策略

基于资源的JSON策略是您附加到资源的策略文档。基于资源的策略的示例包括IAM角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务.

基于资源的策略是位于该服务中的内联策略。你不能用 AWS 基于资源的策略IAM中的托管策略。

访问控制列表 (ACLs)

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。ACLs与基于资源的策略类似，尽管它们不使用JSON策略文档格式。

亚马逊 S3， AWS WAF，Amazon VPC 就是支持的服务示例ACLs。要了解更多信息ACLs，请参阅《亚马逊简单存储服务开发者指南》中的访问控制列表 (ACL) 概述。

其他策略类型

AWS 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界-权限边界是一项高级功能，您可以在其中设置基于身份的策略可以向IAM实体（IAM用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中

的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅《IAM用户指南》中的IAM实体的权限边界。

- 服务控制策略 (SCPs)-SCPs 是指定组织或组织单位 (OU) 的最大权限的JSON策略 AWS Organizations. AWS Organizations 是一项用于对多个进行分组和集中管理的服务 AWS 账户 你的企业拥有的。如果您启用组织中的所有功能，则可以将服务控制策略 (SCPs) 应用于您的任何或所有帐户。SCP限制了成员账户中实体的权限，包括每个 AWS 账户根用户。 有关 Organization SCPs s 和的更多信息，请参阅中的服务控制策略 AWS Organizations 用户指南。
- 会话策略 – 会话策略是当您以编程方式为角色或联合用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅《IAM用户指南》中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解如何做 AWS 决定在涉及多种策略类型时是否允许请求，请参阅IAM用户指南中的策略评估逻辑。

EMR无服务器的工作原理 IAM

在使用IAM管理对 Amazon EMR Serverless 的访问权限之前，请先了解哪些IAM功能可用于 Amazon EMR Serverless。

IAM可以在EMR无服务器中使用的功能

IAM特征	Amazon EMR 无服务器支持
基于身份的策略	是
基于资源的策略	否
策略操作	是
策略资源	是
策略条件密钥	否
ACLs	否
ABAC（策略中的标签）	是

IAM特征	Amazon EMR 无服务器支持
临时凭证	是
主体权限	是
服务角色	否
服务相关角色	是

要全面了解EMR无服务器及其他 AWS 服务适用于大多数IAM功能，请参阅 AWS 《IAM用户指南》IAM中与之配合使用的服务。

无服务器的基于身份的策略 EMR

支持基于身份的策略：是

基于身份的策略是可以附加到身份（例如IAM用户、用户组或角色）的JSON权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅IAM用户指南中的创建IAM策略。

使用IAM基于身份的策略，您可以指定允许或拒绝的操作和资源，以及允许或拒绝操作的条件。您无法在基于身份的策略中指定主体，因为它适用于其附加的用户或角色。要了解可以在JSON策略中使用的所有元素，请参阅IAM用户指南中的IAMJSON策略元素参考。

无服务器基于身份的策略示例 EMR

要查看 Amazon EMR Serverless 基于身份的策略示例，请参阅。无服务器基于身份的策略示例 EMR

Serverless 中EMR基于资源的策略

支持基于资源的策略：否

基于资源的JSON策略是您附加到资源的策略文档。基于资源的策略的示例包括IAM角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务.

要启用跨账户访问，您可以将整个账户或另一个账户中的IAM实体指定为基于资源的策略中的委托人。将跨账户主体添加到基于资源的策略只是建立信任关系工作的一半而已。当委托人和资源处于不同状态时 AWS 账户，可信账户中的IAM管理员还必须向委托人实体（用户或角色）授予访问资源的权限。他们通过将基于身份的策略附加到实体以授予权限。但是，如果基于资源的策略向同一个账户中的主体授予访问权限，则不需要额外的基于身份的策略。有关更多信息，请参阅《IAM用户指南》IAM中的跨账户资源访问权限。

EMR无服务器的策略操作

支持策略操作：是

管理员可以使用 AWS JSON用于指定谁有权访问什么的策略。也就是说，哪个主体 可以对什么资源执行操作，以及在什么条件下执行。

JSON策略`Action`元素描述了可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的同名 AWS API操作。也有一些例外，例如没有匹配API操作的仅限权限的操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行关联操作的权限。

要查看EMR无服务器操作列表，请参阅《服务授权参考》中的 Amazon EMR Serverless 的操作、资源和条件密钥。

EMRServerless 中的策略操作在操作前使用以下前缀。

```
emr-serverless
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
      "emr-serverless:action1",
      "emr-serverless:action2"
         ]
```

要查看 Amazon EMR Serverless 基于身份的策略示例，请参阅。无服务器基于身份的策略示例 EMR

EMR无服务器策略资源

支持策略资源：是

管理员可以使用 AWS JSON用于指定谁有权访问什么的策略。也就是说，哪个主体 可以对什么资源执行操作，以及在什么条件下执行。

ResourceJSON策略元素指定要应用操作的一个或多个对象。语句必须包含 Resource 或 NotResource 元素。最佳做法是，使用资源的 Amazon 资源名称 (ARN) 指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符 (*) 指示语句应用于所有资源。

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

要查看 Amazon EMR 无服务器资源类型及其列表ARNs，请参阅《服务授权参考》中的 Amazon EMR Serverless 定义的资源。要了解您可以为每种资源指定哪些操作，请参阅 Amazon EMR Serverless 的操作、资源和条件键。ARN

要查看 Amazon EMR Serverless 基于身份的策略示例，请参阅。无服务器基于身份的策略示例 EMR

EMR无服务器的策略条件密钥

支持特定于服务的策略条件密钥	否

管理员可以使用 AWS JSON用于指定谁有权访问什么的策略。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

在 Condition 元素（或 Condition 块）中，可以指定语句生效的条件。Condition 元素是可选的。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一条语句中指定多个Condition元素，或者在单个Condition元素中指定多个键， AWS 使用逻辑AND运算对其进行评估。如果您为单个条件键指定多个值， AWS 使用逻辑OR运算评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在资源上标有IAM用户的用户名时，您才能向IAM用户授予访问该资源的权限。有关更多信息，请参阅《IAM用户指南》中的IAM策略元素：变量和标签。

AWS 支持全局条件密钥和特定于服务的条件密钥。要查看全部 AWS 全局条件键，请参见 AWS 《IAM 用户指南》中的全局条件上下文密钥。

要查看 Amazon EMR Serverless 条件密钥列表并了解您可以使用条件密钥的操作和资源，请参阅《服务授权参考》中的 Amazon EMR Serverless 的操作、资源和条件密钥。

所有 Amazon EC2 操作都支持aws:RequestedRegion和ec2:Region条件键。有关更多信息，请参阅示例：限制对特定区域的访问。

EMR无服务器中的访问控制列表 (ACLs)

支持ACLs：否

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。ACLs与基于资源的策略类似，尽管它们不使用JSON策略文档格式。

使用无服务器实现基于属性的访问控制 (ABAC) EMR

支持ABAC（策略中的标签）	是

基于属性的访问控制 (ABAC) 是一种基于属性定义权限的授权策略。In AWS，这些属性称为标签。您可以将标签附加到IAM实体（用户或角色）和许多实体 AWS 资源的费用。为实体和资源添加标签是的第一步。ABAC然后，您可以设计ABAC策略，允许在委托人的标签与他们尝试访问的资源上的标签匹配时进行操作。

ABAC在快速增长的环境中很有用，也有助于解决策略管理变得繁琐的情况。

要基于标签控制访问，您需要使用 aws:ResourceTag/*key-name*、aws:RequestTag/*key-name* 或 aws:TagKeys 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关的更多信息ABAC，请参阅什么是ABAC？ 在《IAM用户指南》中。要查看包含设置步骤的教程ABAC，请参阅IAM用户指南中的使用基于属性的访问控制 (ABAC)。

在EMR无服务器中使用临时证书

支持临时凭证：是

一段时间 AWS 服务 使用临时证书登录时不起作用。欲了解更多信息，包括哪个 AWS 服务 使用临时证书，请参阅 AWS 服务 可以IAM在《IAM用户指南》中使用。

如果您登录，则使用的是临时证书 AWS Management Console 使用除用户名和密码之外的任何方法。例如，当你访问时 AWS 使用贵公司的单点登录 (SSO) 链接，该过程会自动创建临时证书。当您以用户身份登录控制台，然后切换角色时，您还会自动创建临时凭证。有关切换角色的更多信息，请参阅《IAM用户指南》中的切换到角色（控制台）。

您可以使用手动创建临时证书 AWS CLI 或者 AWS API。然后，您可以使用这些临时证书进行访问 AWS. AWS 建议您动态生成临时证书，而不是使用长期访问密钥。有关更多信息，请参阅中的临时安全证书IAM。

无服务器的跨服务主体EMR权限

支持转发访问会话 (FAS)：是

当您使用IAM用户或角色在中执行操作时 AWS，你被视为校长。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS使用委托人的权限调用 AWS 服务，再加上请求的 AWS 服务 向下游服务发出请求。FAS只有当服务收到需要与其他服务进行交互的请求时，才会发出请求 AWS 服务 或需要完成的资源。在这种情况下，您必须具有执行这两个操作的权限。有关提出FAS请求时的政策详情，请参阅转发访问会话。

EMR无服务器的服务角色

支持服务角色	否

无服务器的服务相关角色 EMR

支持服务相关角色	是

有关创建或管理服务相关角色的详细信息，请参阅 AWS 与之配合使用的服务IAM。在表中查找服务相关角色列中包含 Yes 的表。选择是链接以查看该服务的服务相关角色文档。

在无服务器中使用服务相关角色 EMR

Amazon EMR 无服务器使用 AWS Identity and Access Management (IAM) 服务相关角色。服务相关角色是一种独特的角色类型，直接链接到 EMR Serverless。IAM服务相关角色由 EMR Serverless 预定义，包括该服务调用其他角色所需的所有权限 AWS 代表您提供服务。

服务相关角色可以更轻松地设置 EMR Serverless，因为您不必手动添加必要的权限。EMRServerless 定义了其服务相关角色的权限，除非另有定义，否则只有 EMR Serverless 可以担任其角色。定义的权限包括信任策略和权限策略，并且该权限策略不能附加到任何其他IAM实体。

只有在首先删除相关资源后，您才能删除服务相关角色。这可以保护您的EMR无服务器资源，因为您不能无意中删除访问这些资源的权限。

有关支持服务相关角色的其他服务的信息，请参阅 AWS 与之配合使用的服务，IAM然后在 “服务相关角色” 列中查找 “是” 的服务。选择是和链接，查看该服务的服务相关角色文档。

Serverless 的服务相关角色权限 EMR

EMRServerless 使用名为的服务相关角色使其AWSServiceRoleForAmazonEMRServerless能够调用 AWS APIs代表你。

AWSServiceRoleForAmazonEMRServerless 服务相关角色信任以下服务来代入该角色：

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

名为的角色权限策略`AmazonEMRServerlessServiceRolePolicy`允许 EMR Serverless 对指定资源完成以下操作。

> **Note**
>
> 托管政策内容发生变化，因此此处显示的政策可能已过期。在中查看最多的 up-to-date 策略 AmazonEMRServerless ServiceRolePolicy AWS Management Console.

- 操作：`ec2:CreateNetworkInterface`
- 操作：`ec2:DeleteNetworkInterface`
- 操作：`ec2:DescribeNetworkInterfaces`
- 操作：`ec2:DescribeSecurityGroups`
- 操作：`ec2:DescribeSubnets`
- 操作：`ec2:DescribeVpcs`
- 操作：`ec2:DescribeDhcpOptions`
- 操作：`ec2:DescribeRouteTables`
- 操作：`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"
                    ]
                }
            }
        }
    ]
}
```

此角色附加了以下信任策略，以允许 EMR Serverless 主体担任此角色。

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

必须配置权限以允许实IAM体（例如用户、组或角色）创建、编辑或删除服务相关角色。有关更多信息，请参阅《IAM用户指南》中的服务相关角色权限。

为无服务器创建服务相关角色 EMR

您无需手动创建服务相关角色。当您在中创建新的EMR无服务器应用程序时 AWS Management Console （使用 EMR Studio），AWS CLI，或者 AWS API，EMRServerless 会为您创建服务相关角色。必须配置权限以允许实IAM体（例如用户、组或角色）创建、编辑或删除服务相关角色。

要创建 AWSServiceRoleForAmazonEMRServerless 服务相关角色，请使用 IAM

将以下语句添加到需要创建服务相关角色的IAM实体的权限策略中。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:CreateServiceLinkedRole"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/ops.emr-serverless.amazonaws.com/
AWSServiceRoleForAmazonEMRServerless*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "ops.emr-
serverless.amazonaws.com"}}
}
```

如果您删除该服务相关角色，然后需要再次创建，您可以使用相同流程在账户中重新创建此角色。当您创建新的EMR无服务器应用程序时，Ser EMR verless 会再次为您创建服务相关角色。

您还可以使用IAM控制台在 EMRServerless 用例中创建服务相关角色。在 AWS CLI 或者 AWS API，使用服务名称创建服务相关角色。ops.emr-serverless.amazonaws.com有关更多信息，请参阅《IAM用户指南》中的创建服务相关角色。如果您删除了此服务相关角色，可以使用同样的过程再次创建角色。

为 Serverless 编辑服务相关角色 EMR

EMRServerless 不允许您编辑 AWSServiceRoleForAmazonEMRServerless 服务相关角色，因为各种实体可能会引用该角色。你无法编辑 AWS EMR无服务器服务相关角色使用的-owned IAM 策略，因为它包含无服务器所需的所有必要权限EMR。但是，您可以使用编辑角色的描述IAM。

要编辑 AWSServiceRoleForAmazonEMRServerless 服务相关角色的描述，请使用 IAM

将以下语句添加到需要编辑服务相关角色描述的IAM实体的权限策略中。

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

有关更多信息，请参阅《IAM用户指南》中的编辑服务相关角色。

删除 Serverless 的EMR服务相关角色

如果不再需要使用某个需要服务相关角色的功能或服务，我们建议您删除该角色。这样您就不会有未被主动监控或维护的未使用实体。但是，必须先删除所有区域中的所有 EMR Serverless 应用程序，然后才能删除服务相关角色。

> **Note**
>
> 如果您尝试删除与该角色关联的资源时，EMRServerless 服务正在使用该角色，则删除可能会失败。如果发生这种情况，请等待几分钟后重试。

要删除 AWSServiceRoleForAmazonEMRServerless 服务相关角色，请使用 IAM

将以下语句添加到需要删除服务相关角色的IAM实体的权限策略中。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/ops.emr-serverless.amazonaws.com/
AWSServiceRoleForAmazonEMRServerless*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "ops.emr-
serverless.amazonaws.com"}}
}
```

使用手动删除服务相关角色 IAM

使用IAM控制台， AWS CLI，或者 AWS API删除 AWSServiceRoleForAmazonEMRServerless 服务相关角色。有关更多信息，请参阅《IAM用户指南》中的删除服务相关角色。

EMR无服务器服务相关角色支持的区域

EMRServerless 支持在提供服务的所有区域中使用服务相关角色。有关更多信息，请参阅 AWS 区域和终端节点。

Amazon EMR Serverless 的 Job 运行时角色

您可以指定EMR无服务器作业在代表您调用其他服务时可以承担的IAM角色权限。这包括访问任何数据源、目标以及其他 Amazon S3 AWS 诸如 Amazon Redshift 集群和 DynamoDB 表之类的资源。要了解有关如何创建角色的更多信息，请参阅创建作业运行时角色。

运行时策略示例

您可以将运行时策略（如下所示）附加到作业运行时角色。以下作业运行时策略允许：

- 阅读带EMR示例的 Amazon S3 存储桶访问权限。
- 对 S3 存储桶的完全访问权限。
- 创建和读取访问权限 AWS Glue 数据目录。

添加对其他人的访问权限 AWS 像 DynamoDB 这样的资源，在创建运行时角色时，你需要在策略中包含这些资源的权限。

```
{
    "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:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-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": ["*"]
        }
    ]
}
```

传递角色权限

您可以将IAM权限策略附加到用户的角色，以允许该用户仅传递已批准的角色。这允许管理员控制哪些用户可以将特定的作业运行时角色传递给 EMR Serverless 作业。要了解有关设置权限的更多信息，请参阅向用户授予将角色传递给的权限 AWS 服务。

以下是允许将作业运行时角色传递给 EMR Serverless 服务主体的示例策略。

```
{
     "Effect": "Allow",
     "Action": "iam:PassRole",
     "Resource": "arn:aws:iam::1234567890:role/JobRuntimeRoleForEMRServerless",
        "Condition": {
                "StringLike": {
                    "iam:PassedToService": "emr-serverless.amazonaws.com"
                }
            }
}
```

EMR无服务器的用户访问策略示例

您可以根据您希望每个用户在与EMR无服务器应用程序交互时执行的操作，为用户设置精细的策略。以下策略是可能有助于为用户设置正确权限的示例。本节仅重点介绍EMR无服务器策略。有关 EMR Studio 用户策略的示例，请参阅配置 EMR Studio 用户权限。有关如何向IAM用户（委托人）附加策略的信息，请参阅IAM用户指南中的管理IAM策略。

高级用户政策

要向 EMR Serverless 授予所有必需的操作，请创建`AmazonEMRServerlessFullAccess`策略并将其附加到所需的IAM用户、角色或组。

以下是允许高级用户创建和修改EMR无服务器应用程序，以及执行其他操作（例如提交和调试作业）的策略示例。它揭示了 EMR Serverless 需要对其他服务执行的所有操作。

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

当您启用与您的网络连接时VPC，EMR无服务器应用程序会创建 Amazon EC2 弹性网络接口 (ENIs) 来与VPC资源通信。以下策略确保所有新内容EC2ENIs都只能在EMR无服务器应用程序的上下文中创建。

> **Note**
>
> 我们强烈建议您设置此政策，以确保用户EC2ENIs只能在启动EMR无服务器应用程序的情况下进行创建。

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

如果要限制对某些子网的EMR无服务器访问，则可以使用标记条件标记每个子网。此IAM策略可确保EMR无服务器应用程序只能EC2ENIs在允许的子网内创建。

```
{
    "Sid": "AllowEC2ENICreationInSubnetAndSecurityGroupWithEMRTags",
    "Effect": "Allow",
    "Action": [
        "ec2:CreateNetworkInterface"
    ],
    "Resource": [
        "arn:aws:ec2:*:*:subnet/*",
        "arn:aws:ec2:*:*:security-group/*"
    ],
    "Condition": {
        "StringEquals": {
            "aws:ResourceTag/KEY": "VALUE"
        }
    }
}
```

Important

如果您是创建第一个应用程序的管理员或高级用户，则必须配置权限策略以允许您创建EMR无服务器服务相关角色。要了解更多信息，请参阅 在无服务器中使用服务相关角色 EMR。

以下IAM策略允许您为账户创建EMR无服务器服务相关角色。

```
{
```

```
   "Sid":"AllowEMRServerlessServiceLinkedRoleCreation",
   "Effect":"Allow",
   "Action":"iam:CreateServiceLinkedRole",
   "Resource":"arn:aws:iam::account-id:role/aws-service-role/ops.emr-
serverless.amazonaws.com/AWSServiceRoleForAmazonEMRServerless"
}
```

数据工程师政策

以下是一个策略示例，它允许用户拥有EMR无服务器应用程序的只读权限，以及提交和调试作业的能力。请记住，由于此策略不会明确拒绝操作，因此仍可使用其它策略声明来授予对指定操作的访问权限。

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

使用标签进行访问控制

您可以使用标签条件进行精细的访问控制。例如，您可以限制来自一个团队的用户，使他们只能向标有团队名称的EMR无服务器应用程序提交作业。

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

基于标签的访问控制策略

您可以使用基于身份的策略中的条件，根据标签控制对应用程序和作业运行的访问权限。

以下示例演示了在EMR无服务器条件键中使用条件运算符的不同场景和方法。这些IAM政策声明仅用于演示目的，不应在生产环境中使用。可通过多种方式来组合策略声明，以根据要求授予和拒绝权限。有关规划和测试IAM策略的更多信息，请参阅《IAM用户指南》。

> **Important**
>
> 一个重要注意事项是，应该明确拒绝添加标签操作的权限。这可以防止用户标记资源，从而为其授予您不打算授予的权限。如果资源的标记操作未被拒绝，用户可以修改标记并规避基于标签的策略意图。有关拒绝添加标签操作的策略示例，请参阅拒绝添加和删除标签的访问权限。

以下示例演示了基于身份的权限策略，这些策略用于控制EMR无服务器应用程序允许的操作。

仅允许带特定标签值资源上的操作

在以下策略示例中，StringEquals条件运算符尝试dev与标签部门的值相匹配。如果标签部门尚未添加到应用程序中，或者不包含该值dev，则该策略不适用，并且此策略不允许执行这些操作。如果没有其他策略声明允许这些操作，则用户只能使用带有此值的此标签的应用程序。

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

您也可以使用条件运算符指定多个标签值。例如，要允许对department标签包含值dev或的应用程序执行操作test，可以将前面示例中的条件块替换为以下内容。

```
"Condition": {
        "StringEquals": {
          "emr-serverless:ResourceTag/department": ["dev", "test"]
        }
      }
```

创建资源时需要标签

在下面的示例中，需要在创建应用程序时应用标签。

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

```
      "Action": [
        "emr-serverless:CreateApplication"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "emr-serverless:RequestTag/department": "dev"
        }
      }
    }
  ]
}
```

以下策略声明仅允许用户在应用程序具有可以包含任何值的department标签时创建应用程序。

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

拒绝添加和删除标签的访问权限

此策略禁止用户在EMR无服务器应用程序上添加或删除标签值不dev为的department标签。

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

```
        "emr-serverless:TagResource",
        "emr-serverless:UntagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "emr-serverless:ResourceTag/department": "dev"
        }
      }
    }
  ]
}
```

无服务器基于身份的策略示例 EMR

默认情况下，用户和角色无权创建或修改 Amazon EMR Serverless 资源。他们也无法通过使用来执行任务 AWS Management Console, AWS Command Line Interface (AWS CLI)，或 AWS API。要授予用户对其所需资源执行操作的权限，IAM管理员可以创建IAM策略。然后，管理员可以将IAM策略添加到角色中，用户可以代入这些角色。

要了解如何使用这些示例策略文档创建IAM基于身份的JSON策略，请参阅IAM用户指南中的创建IAM策略。

有关 Amazon EMR Serverless 定义的操作和资源类型（包括每种资源类型的格式）的详细信息，请参阅《服务授权参考》中的 Amazon EMR Serverless 的操作、资源和条件密钥。ARNs

主题

- 策略最佳实践
- 允许用户查看他们自己的权限

策略最佳实践

> **Note**
>
> EMRServerless 不支持托管策略，因此下面列出的第一种做法不适用。

基于身份的策略决定是否有人可以在您的账户中创建、访问或删除亚马逊EMR无服务器资源。这些操作可能会使您付出代价 AWS 账户。 创建或编辑基于身份的策略时，请遵循以下准则和建议：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用 AWS 为许多常见用例授予权限的托管策略。它们在你的 AWS 账户。 我们建议您通过定义来进一步减少权限 AWS 特定于您的用例的客户托管政策。有关更多信息，请参阅 AWS 托管策略或 AWS 《IAM 用户指南》 中工作职能的托管策略。
- 应用最低权限权限-使用IAM策略设置权限时，仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用应用权限IAM的更多信息，请参阅IAM用户指南IAM中的策略和权限。
- 使用IAM策略中的条件进一步限制访问权限-您可以在策略中添加条件以限制对操作和资源的访问权限。例如，您可以编写一个策略条件来指定所有请求都必须使用发送SSL。如果通过特定条件使用服务操作，则也可以使用条件来授予对服务操作的访问权限 AWS 服务之外的压缩算法（例如 AWS CloudFormation。 有关更多信息，请参阅《IAM用户指南》中的IAMJSON策略元素：条件。
- 使用 A IAM ccess Analyzer 验证您的IAM策略以确保权限的安全性和功能性 — A IAM ccess Analyzer 会验证新的和现有的策略，以便策略符合IAM策略语言 (JSON) 和IAM最佳实践。IAMAccess Analyzer 提供了 100 多项策略检查和可行的建议，可帮助您制定安全和实用的策略。有关更多信息，请参阅《IAM用户指南》中的 IAMAccess Analyzer 策略验证。
- 需要多因素身份验证 (MFA)-如果您的场景需要IAM用户或 root 用户 AWS 账户，请打开MFA以提高安全性。要要求MFA何时调用API操作，请在策略中添加MFA条件。有关更多信息，请参阅《IAM用户指南》中的配置MFA受保护的API访问权限。

有关中最佳做法的更多信息IAM，请参阅《IAM用户指南》IAM中的安全最佳实践。

允许用户查看他们自己的权限

此示例说明如何创建允许IAM用户查看附加到其用户身份的内联和托管策略的策略。此策略包括通过控制台或以编程方式使用控制台完成此操作的权限 AWS CLI 或者 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": "*"
        }
    ]
}
```

Amazon EMR 无服务器更新至 AWS 托管策略

查看有关更新的详细信息 AWS 自从该服务开始跟踪这些更改以来，Amazon EMR Serverless 的托管策略。要获取有关此页面变更的自动提醒，请在 Amazon EMR Serverless 文档历史记录页面上订阅 RSS Feed。

更改	描述	日期
A mazonEMRServerless ServiceRolePolicy -更新现有政策	Amazon EMR Serverless 在 A mazonEMRServerless ServiceRolePolicy 策略中`EC2PolicyStatement` 添加了新的`SidCloudWatchPolicyStatement` 和。	2024 年 1 月 25 日

更改	描述	日期
A mazonEMRServerless ServiceRolePolicy -更新现有政策	Amazon EMR Serverless 添加了新的权限，允许 Amazon EMR Serverless 发布命名空间中 v CPU 使用情况的汇总账户指标。"AWS/Usage"	2023 年 4 月 20 日
Amazon EMR Serverless 开始跟踪变更	Amazon EMR Serverless 开始跟踪其变更 AWS 托管策略。	2023 年 4 月 20 日

对 Amazon EMR 无服务器身份和访问进行故障排除

使用以下信息来帮助您诊断和修复在使用 Amazon EMR Serverless 和IAM时可能遇到的常见问题。

主题

- 我无权在 Amazon EMR Serverless 中执行任何操作
- 我无权执行 iam：PassRole
- 我想允许我以外的人进入 AWS 访问我的 Amazon EMR 无服务器资源的账户

我无权在 Amazon EMR Serverless 中执行任何操作

如果 AWS Management Console 告诉您您无权执行某项操作，那么您必须联系管理员寻求帮助。管理员是指提供用户名和密码的人员。

当 `mateojackson` 用户尝试使用控制台查看有关虚构 *`my-example-widget`* 资源的详细信息，但不拥有虚构 `emr-serverless:`*`GetWidget`* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: emr-
serverless:GetWidget on resource: my-example-widget
```

在这种情况下，Mateo 请求他的管理员更新其策略，以允许他使用 `emr-serverless:`*`GetWidget`* 操作访问 *`my-example-widget`* 资源。

我无权执行 iam：PassRole

如果您收到错误消息，指出您无权执行该`iam:PassRole`操作，则必须更新您的策略以允许您将角色传递给 Amazon EMR Serverless。

一段时间 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为的IAM用户`marymajor`尝试使用控制台在 Amazon EMR Serverless 中执行操作时，会出现以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

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

在这种情况下，必须更新 Mary 的策略以允许她执行 `iam:PassRole` 操作。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我想允许我以外的人进入 AWS 访问我的 Amazon EMR 无服务器资源的账户

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以担任角色。对于支持基于资源的策略或访问控制列表 (ACLs) 的服务，您可以使用这些策略向人们授予访问您的资源的权限。

要了解更多信息，请参阅以下内容：

- 要了解 Amazon EMR Serverless 是否支持这些功能，请参阅Amazon EMR Serverless 中的身份和访问管理 (IAM)。
- 要了解如何提供对您的资源的访问权限 AWS 账户 您拥有的，请参阅向其他IAM用户提供访问权限 AWS 账户 您在《IAM用户指南》中拥有的内容。
- 了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅提供访问权限 AWS 账户 《IAM用户指南》中由第三方所有。
- 要了解如何通过联合身份验证提供访问权限，请参阅《用户指南》中的向经过外部身份验证的用户提供访问权限（联合身份验证）。IAM
- 要了解使用角色和基于资源的策略进行跨账户访问的区别，请参阅IAM用户指南IAM中的跨账户资源访问权限。

将EMR无服务器与 AWS Lake Formation 用于精细的访问控制

概述

随着 Amazon 7.2.0 及更高版本的EMR发布，您可以利用 AWS Lake Formation 对 S3 支持的数据目录表应用精细的访问控制。此功能允许您配置表、行、列和单元格级别的访问控制 read 在您的 Amazon EMR 无服务器 Spark 任务中进行查询。要为 Apache Spark 批处理作业和交互式会话配置精细的访问控制，请使用 Studio。EMR请参阅以下章节，详细了解 Lake Formation 以及如何将其与 EMR Serverless 配合使用。

将 Amazon EMR Serverless 与 AWS Lake Formation 会产生额外费用。有关更多信息，请参阅 Amazon EMR 定价。

EMR无服务器的工作原理 AWS Lake Formation

使用 EMR Serverless 和 Lake Formation 可以让你对每个 Spark 作业强制执行一层权限，以便在EMR无服务器执行作业时应用 Lake Formation 权限控制。EMRServerless 使用 Spark 资源配置文件创建两个配置文件以有效执行作业。用户配置文件执行用户提供的代码，而系统配置文件则强制执行 Lake Formation 策略。有关更多信息，请参阅什么是 AWS Lake Formation以及注意事项和限制。

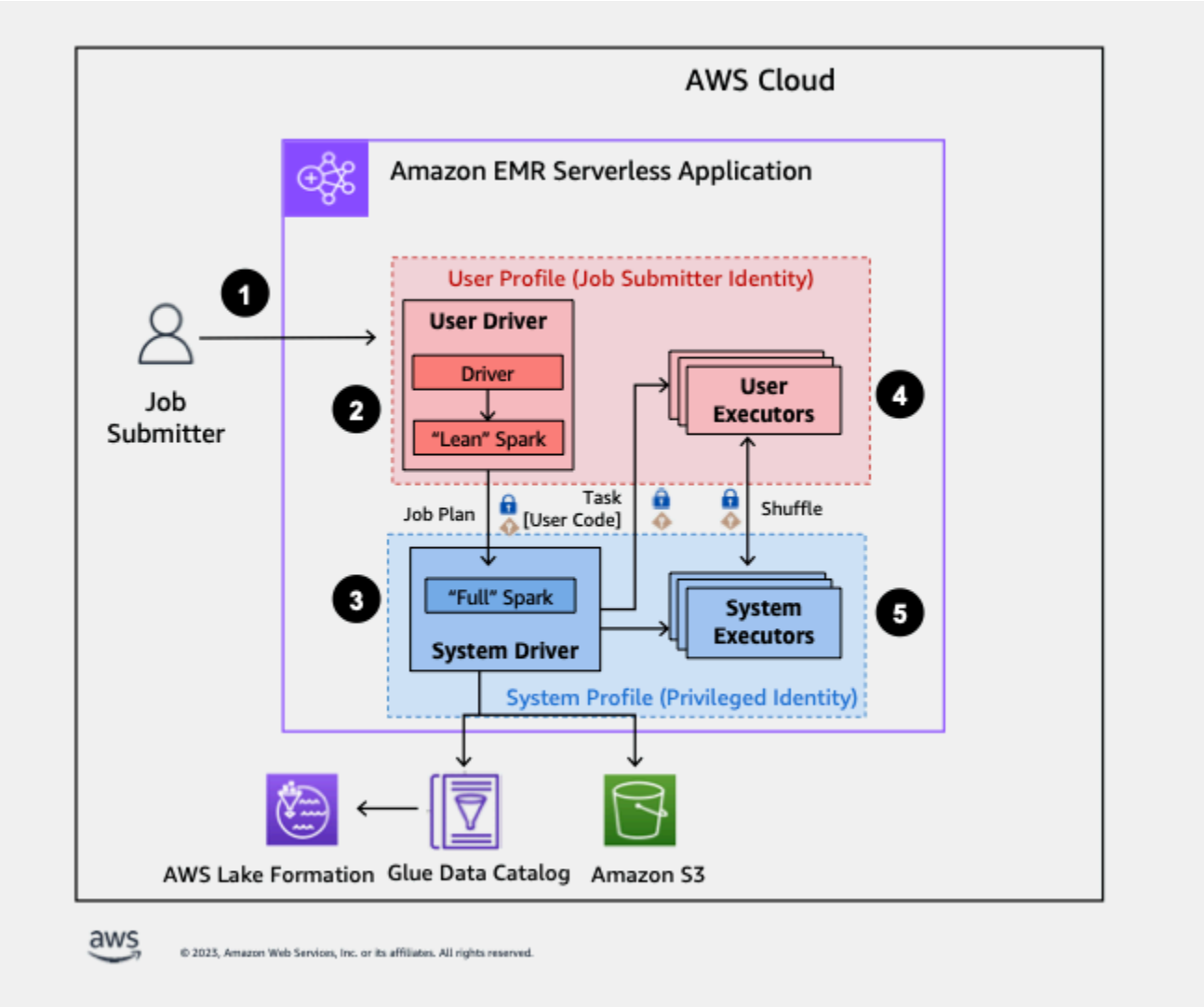
当你在 Lake Formation 中使用预先初始化的容量时，我们建议你至少有两个 Spark 驱动程序。每个支持 Lake Formation 的作业都使用两个 Spark 驱动程序，一个用于用户配置文件，一个用于系统配置文件。为了获得最佳性能，与不使用 Lake Formation 相比，在支持 Lake Formation 的工作中，你使用的司机数量应该是原来的两倍。

在 EMR Serverless 上运行 Spark 作业时，还必须考虑动态分配对资源管理和集群性能的影响。每个资源配置文件`spark.dynamicAllocation.maxExecutors`的最大执行者数量的配置适用于用户和系统执行者。如果您将该数字配置为等于允许的最大执行者数，则您的作业运行可能会停滞不前，因为一种类型的执行器使用了所有可用资源，这会在您运行作业时阻止其他执行器。

因此，您不会耗尽资源，EMRServerless 将每个资源配置文件默认的最大执行者数设置为该`spark.dynamicAllocation.maxExecutors`值的 90%。如果使用介于 0 和 1 之间的值`spark.dynamicAllocation.maxExecutorsRatio`进行指定，则可以覆盖此配置。此外，您还可以配置以下属性来优化资源分配和整体性能：

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

以下简要概述了EMR无服务器如何访问受 Lake Formation 安全策略保护的数据。



1. 用户将 Spark 作业提交给一个 AWS Lake Formation启用了EMR无服务器应用程序。
2. EMRServerless 将任务发送给用户驱动程序，并在用户配置文件中运行该作业。用户驱动程序运行精简版的 Spark，该版本无法启动任务、请求执行器、访问 S3 或 Glue 目录。它制定了工作计划。
3. EMRServerless 会设置第二个名为系统驱动程序的驱动程序，并在系统配置文件中运行该驱动程序（使用特权身份）。EMRServerless 在两个驱动程序之间设置了用于通信的加密TLS通道。用户驱动程序使用该通道将工作计划发送给系统驱动程序。系统驱动程序不运行用户提交的代码。它运行完整 Spark 并与 S3 和数据目录通信以进行数据访问。它请求执行者并将Job Plan编译成一系列执行阶段。
4. EMR然后，Serverless 使用用户驱动程序或系统驱动程序在执行器上运行阶段。任何阶段的用户代码都只能在用户配置文件执行器上运行。

5. 从受保护的数据目录表中读取数据的阶段 AWS Lake Formation 或者那些应用安全筛选器的过滤器被委托给系统执行者。

在亚马逊中启用 Lake Formation EMR

要启用 Lake Formation，在创建EMR无服务器应用程序时，必须将运行时配置参数设置`spark.emr-serverless.lakeformation.enabled`为`spark-defaults`分类`true`下。

```
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"
```

在 EMR Studio 中创建新应用程序时，也可以启用 Lake Formation。选择 “使用 Lake Formation” 进行精细的访问控制，可在 “其他配置” 下找到。

当您@@ 将 Lake Formation 与 EMR Serverless 结合使用时，默认情况下会启用工作器间加密，因此您不必再次明确启用工作器间加密。

为 Spark 作业启用 Lake Formation

要为单个 Spark 作业启用 Lake Formation，请在使用时`spark.emr-serverless.lakeformation.enabled`将其设置为 true `spark-submit`。

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

Job 运行时角色IAM权限

Lake Formation 权限控制访问权限 AWS Glue 数据目录资源、Amazon S3 位置以及这些位置的基础数据。IAM权限控制对 Lake Formation 的访问权限和 AWS Glue APIs 和资源。尽管您可能拥有 Lake Formation 访问数据目录 (SELECT) 中表的权限，但如果您没有该操作的IAM权限，则`glue:Get*`API操作将失败。

以下是如何提供访问 S3 中的脚本、将日志上传到 S3 的IAM权限的策略示例， AWS Glue API 权限和访问 Lake Formation 的权限。

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ScriptAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
            "arn:aws:s3:::*.DOC-EXAMPLE-BUCKET/scripts",
            "arn:aws:s3:::*.DOC-EXAMPLE-BUCKET/*" ]
        },
        {
         "Sid": "LoggingAccess",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
               "arn:aws:s3:::DOC-EXAMPLE-BUCKET/logs/*"
            ]
        },
        {
            "Sid": "GlueCatalogAccess",
            "Effect": "Allow",
            "Action": [
                 "glue:Get*",
                 "glue:Create*",
                 "glue:Update*"
            ],
            "Resource": ["*"]
        },
        {
            "Sid": "LakeFormationAccess",
            "Effect": "Allow",
            "Action": [
             "lakeformation:GetDataAccess"
             ],
            "Resource": ["*"]
        }
    ]
```

```
}
```

为作业运行时角色设置 Lake Formation 权限

首先，在 Lake Formation 中注册 Hive 桌子的位置。然后在所需的表上为您的作业运行时角色创建权限。有关 Lake Formation 的更多详情，请参阅什么是 AWS Lake Formation？ 在 AWS Lake Formation 开发者指南。

设置 Lake Formation 权限后，您可以在 Amazon EMR Serverless 上提交 Spark 任务。有关 Spark 任务的更多信息，请参阅 Spark 示例。

提交作业运行

完成 Lake Formation 授权的设置后，你可以在 EMRServerless 上提交 Spark 作业。要运行 Iceberg 作业，您必须提供以下`spark-submit`属性。

```
--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
```

开放表格格式支持

亚马逊EMR版本 7.2.0 支持基于 Lake Formation 的精细访问控制。EMR无服务器支持 Hive 和 Iceberg 表类型。下表描述了所有支持的操作。

操作	Hive	Iceberg
DDL命令	仅限IAM角色权限	仅限IAM角色权限
递增查询	不适用	完全支持
时间旅行查询	不适用于此表格格式	完全支持
元数据表	不适用于此表格格式	支持，但某些表格处于隐藏状态。有关更多信息，请参阅注意事项和限制。

操作	Hive	Iceberg
DML INSERT	仅限IAM权限	仅限IAM权限
DML UPDATE	不适用于此表格格式	仅限IAM权限
DML DELETE	不适用于此表格格式	仅限IAM权限
读取操作	完全支持	完全支持
存储过程	不适用	支持，但register_table 和除外migrate。有关更多信息，请参阅注意事项和限制。

注意事项和限制

将 Lake Formation 与EMR无服务器一起使用时，请考虑以下注意事项和限制。

> **Note**
>
> 当你在 EMR Serverless 上为 Spark 作业启用 Lake Formation 时，该作业会启动系统驱动程序和用户驱动程序。如果您在启动时指定了预初始化的容量，则从预初始化的容量中配置驱动程序，系统驱动程序的数量等于您指定的用户驱动程序的数量。如果您选择 “按需容量”，EMRServerless 将启动除用户驱动程序之外的系统驱动程序。要估算与 EMR Serverless with Lake Formation 任务相关的成本，请使用 AWS Pricing Calculator.

带有 Lake F EMR ormation 的 Amazon Serverless 适用于所有支持的EMR无服务器区域，但以下区域除外 AWS GovCloud （美国东部）和 AWS GovCloud （美国西部）。

- Amazon EMR Serverless 仅支持通过 Lake Formation 对 Apache Hive 和 Apache Iceberg 表进行精细访问控制。Apache Hive 格式包括 Parquet ORC、和 xSV。
- 支持 Lake Formation 的应用程序不支持使用自定义的EMR无服务器镜像。
- 你无法关闭 Lake F DynamicResourceAllocation ormation 的工作。
- 你只能将 Lake Formation 与 Spark 任务一起使用。
- EMR带有 Lake Formation 的无服务器在整个作业中仅支持单个 Spark 会话。

- EMR带有 Lake Formation 的 Serverless 仅支持通过资源链接共享的跨账户表格查询。
- 不支持以下项：
  - 弹性分布式数据集 (RDD)
  - Spark 直播
  - 在 Lake Formation 获得权限的情况下写作
  - 嵌套列的访问控制
- EMRServerless 会阻止可能破坏系统驱动程序完全隔离的功能，包括以下功能：
  - UDTsiveUDFs、H 以及任何涉及自定义类的用户定义函数
  - 自定义数据源
  - 为 Spark 扩展、连接器或元数据仓提供额外的罐子
  - `ANALYZE TABLE` 命令
- 强制执行访问控制`EXPLAIN PLAN`和诸如 “`DESCRIBE TABLE`不公开受限信息” 之类的DDL操作。
- EMRServerless 限制了在支持 Lake Formation 的应用程序上访问系统驱动程序 Spark 日志。由于系统驱动程序在运行时具有更多的访问权限，因此系统驱动程序生成的事件和日志可能包含敏感信息。为了防止未经授权的用户或代码访问这些敏感数据，EMRServerless 禁用了对系统驱动程序日志的访问。要进行故障排除，请联系 AWS 支持。
- 如果您在 Lake Formation 中注册了表位置，则无论EMR无服务器作业运行时角色的IAM权限如何，数据访问路径都会通过 Lake Formation 存储的凭据。如果您错误配置了向表位置注册的角色，则提交的使用该角色对表位置具有 S3 IAM 权限的任务将失败。
- 写入 Lake Formation 表使用IAM权限而不是 Lake Formation 授予的权限。如果您的作业运行时角色具有必要的 S3 权限，则可以使用它来运行写入操作。

以下是使用 Apache Iceberg 时的注意事项和限制：

- 你只能将 Apache Iceberg 与会话目录一起使用，不能使用任意命名的目录。
- 在 Lake Formation 中注册的 Iceberg 表仅支持元数据表`history`、`metadata_log_entries`、`snapshots`、`files`、`manifests`、和`refs`。Amazon 会EMR隐藏可能包含敏感数据的列，例如`partitions`、`path`和。`summaries`此限制不适用于未在 Lake Formation 中注册的 Iceberg 表。
- 您未在 Lake Formation 中注册的表支持所有 Iceberg 存储过程。任何表都不支持`register_table`和`migrate`过程。
- 我们建议您使用 Iceberg DataFrameWriter V2 而不是 V1。

故障排除

有关疑难解答的解决方案，请参阅以下部分。

日志记录

EMR无服务器使用 Spark 资源配置文件来拆分任务执行。EMRServerless 使用用户配置文件来运行您提供的代码，而系统配置文件则强制执行 Lake Formation 策略。您可以访问以用户配置文件形式运行的任务的日志。

实时用户界面和 Spark 历史服务器

Live UI 和 Spark History Server 包含根据用户配置文件生成的所有 Spark 事件以及由系统驱动程序生成的经过编辑的事件。

您可以在 “E xecutor s” 选项卡中查看用户和系统驱动程序中的所有任务。但是，日志链接仅适用于用户个人资料。此外，还会从 Live UI 中删除一些信息，例如输出记录的数量。

由于 Lake Formation 权限不足，作业失败

确保您的作业运行时角色具有运行权限SELECT以及DESCRIBE您正在访问的表的权限。

RDD执行失败的 Job

EMRServerless 目前不支持对启用 Lake Formation 的作业进行弹性分布式数据集 (RDD) 操作。

无法访问 Amazon S3 中的数据文件

确保您已在 Lake Formation 中注册了数据湖的位置。

安全验证异常

EMRServerless 检测到安全验证错误。联系人 AWS 对援助的支持。

共享 AWS Glue 数据目录和跨账户的表格

您可以跨账户共享数据库和表，但仍然可以使用 Lake Formation。有关更多信息，请参阅 Lake Formation 中的跨账户数据共享和如何共享 AWS 使用 Glue 跨账户数据目录和表格 AWS Lake Formation?.

工作者间加密

在 Amazon 6.15.0 及更高EMR版本中，您可以在 Spark 作业运行中启用工作人员之间的相互TLS加密通信。启用后，EMRServerless 会自动为任务运行中配置的每个工作程序生成和分发唯一的证书。当这些工作人员进行通信以交换控制消息或传输 shuffle 数据时，他们会建立相互TLS连接并使用配置的证书来验证彼此的身份。如果工作人员无法验证其他证书，则TLS握手失败，EMRServerless 将中止他们之间的连接。

如果您将 Lake Formation 与 EMR Serverless 配合使用，则默认情况下会启用双向TLS加密。

在EMR无服务器上启用双向TLS加密

要在 spark 应用程序上启用相互TLS加密，请在创建EMR无服务器应用程序时`spark.ssl.internode.enabled`将其设置为 true。如果你使用的是 AWS 控制台要创建EMR无服务器应用程序，请选择使用自定义设置，然后展开应用程序配置，然后输入您的`runtimeConfiguration`。

```
aws emr-serverless create-application \
--release-label emr-6.15.0 \
--runtime-configuration '{
  "classification": "spark-defaults",
  "properties": {"spark.ssl.internode.enabled": "true"}
}' \
--type "SPARK"
```

如果要为单个 Spark 作业运行启用相互TLS加密，请在使用时设置`spark.ssl.internode.enabled`为 true `spark-submit`。

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

使用EMR无服务器进行数据保护的 Secrets Manager

AWS Secrets Manager 是一项密钥存储服务，可用于保护数据库凭据、API密钥和其他机密信息。然后，在你的代码中，你可以用调API用 Secrets Manager 来替换硬编码的凭据。这有助于确保密钥不会被检查你的代码的人泄露，因为密码不存在。有关概述，请参阅 AWS Secrets Manager 用户指南。

Secrets Manager 使用以下方法加密机密 AWS Key Management Service 钥匙。有关更多信息，请参阅中的秘密加密和解密 AWS Secrets Manager 用户指南。

此外，您还可以配置 Secrets Manager 以根据指定的计划自动轮换密钥。这使您能够将长期密钥替换为短期密钥，这有助于显著减少泄露风险。有关更多信息，请参阅旋转 AWS Secrets Manager里面的秘密 AWS Secrets Manager 用户指南。

Amazon EMR Serverless 与 AWS Secrets Manager 这样你就可以将数据存储在 Secrets Manager 中并在配置中使用密钥 ID。

EMR无服务器如何使用机密

当您将数据存储在 Secrets Manager 中并在EMR无服务器配置中使用密钥 ID 时，您不会以纯文本形式将敏感配置数据传递给 EMR Serverless，也不会将其暴露给外部。APIs如果您指明键值对包含您存储在 Secrets Manager 中的密钥的密钥 ID，则 EMR Serverless 会在向正在运行的作业的工作人员发送配置数据时检索该密钥。

要表示配置的键值对包含对存储在 Secrets Manager 中的密钥的引用，请将`EMR.secret@`注释添加到配置值中。对于任何带有机密 ID 注释的配置属性，EMRServerless 都会调用 Secrets Manager 并在任务执行时解析密钥。

如何创建密钥

要创建密钥，请按照创建密钥中的步骤进行操作 AWS Secrets Manager 秘密在 AWS Secrets Manager 用户指南。在步骤 3 中，选择 Plaintext 字段以输入您的敏感值。

在配置分类中提供密钥

以下示例说明如何在配置分类中提供密钥`StartJobRun`。如果要在应用程序级别为 Secrets Manager 配置分类，请参阅EMR无服务器的默认应用程序配置。

在示例中，*`SecretName`*替换为要检索的密钥的名称。包括连字符，然后是 Secrets Manager 在密钥末尾添加的六个字符。ARN有关更多信息，请参阅 如何创建密钥。

本节内容

- 指定秘密引用-Spark
- 指定秘密引用-Hive

指定秘密引用-Spark

Example — 在外部 Hive 元数据仓配置中为 Spark 指定秘密引用

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

```
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
  --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://DOC-EXAMPLE-BUCKET/scripts/spark-jdbc.py",
            "sparkSubmitParameters": "--jars s3://DOC-EXAMPLE-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://DOC-EXAMPLE-BUCKET/spark/logs/"
        }
    }
}'
```

Example — 为分类中的外部 Hive 元数据仓配置指定秘密引用 **spark-defaults**

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

指定秘密引用-Hive

Example — 在 Hive 的外部 Hive 元数据仓配置中为 Hive 指定秘密引用

```
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.exec.scratchdir=s3://DOC-EXAMPLE-BUCKET/emr-
serverless-hive/hive/scratch
                    --hiveconf hive.metastore.warehouse.dir=s3://DOC-EXAMPLE-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 — 为分类中的外部 Hive 元数据仓配置指定秘密引用 **hive-site**

```
{
    "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"
    }
}
```

向 EMR Serverless 授予检索密钥的访问权限

要允许 EMR Serverless 从 Secrets Manager 中检索密钥值，请在创建密钥时向其添加以下策略声明。您必须使用客户管理的密钥创建您的密KMS钥，EMRServerless 才能读取密钥值。有关更多信息，请参阅《》中的KMS密钥权限 AWS Secrets Manager 用户指南。

在以下策略中，*applicationId*替换为应用程序的 ID。

机密的资源政策

您必须在密钥的资源策略中包含以下权限 AWS Secrets Manager 允许 EMR Serverless 检索机密值。为确保只有特定的应用程序才能检索此密钥，您可以选择在策略中指定EMR无服务器应用程序 ID 作为条件。

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

```
  ]
}
```

使用以下策略为客户管理的密钥创建您的密钥 AWS Key Management Service (AWS KMS) 密钥：

客户管理政策 AWS KMS 钥匙

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

轮换秘密

轮换是指定期更新密钥。你可以配置 AWS Secrets Manager 按您指定的时间表自动轮换密钥。这样，您就可以用短期机密替换长期机密。这有助于降低泄露的风险。EMR当作业转换到运行状态时，Serverless 会从带注释的配置中检索密钥值。如果您或某个进程在 Secrets Manager 中更新了密钥值，则必须提交新作业，这样该任务才能获取更新的值。

> **Note**
>
> 已经处于运行状态的作业无法获取更新的密钥值。这可能会导致任务失败。

将 Amazon S3 访问权限授予与EMR无服务器配合使用

EMR无服务器的 S3 访问权限授予概述

在 Amazon 6.15.0 及更高EMR版本中，Amazon S3 访问授权提供了一种可扩展的访问控制解决方案，您可以使用该解决方案来增强从 Serverless 对您的 Amazon S3 数据的访问权限。EMR如果您的 S3 数据有复杂或大规模的权限配置，则可以使用 Access Grants 来扩展用户、角色和应用程序的 S3 数据权限。

使用 S3 访问权限授予对Amazon S3数据的访问权限，超出运行时IAM角色或附加到有权访问您的EMR无服务器应用程序的身份的角色所授予的权限。

有关更多信息，请参阅《亚马逊管理指南》EMR中的 “使用亚马逊的 S3 访问授权EMR管理访问权限”和《亚马逊简单存储服务用户指南》中的 “使用 S3 访问权限管理访问权限”。

本节介绍如何启动使用 S3 访问权限来提供对 Amazon S3 中数据的访问权限的EMR无服务器应用程序。有关将 S3 访问权限授予用于其他 Amazon EMR 部署的步骤，请参阅以下文档：

- 在 Amazon 上使用 S3 访问授权 EMR
- 在 Amazon EMR 上使用 S3 访问授权 EKS

使用用于数据管理的 S3 访问权限启动EMR无服务器应用程序

您可以在EMR无服务器上启用 S3 访问权限授予并启动 Spark 应用程序。当您的应用程序请求获取 S3 数据时，Amazon S3 会提供限定于特定存储桶、前缀或对象的临时凭证。

1. 为您的EMR无服务器应用程序设置任务执行角色。包括运行 Spark 任务和使用 S3 访问IAM权限所需的权限，`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**
>
> 如果您为任务执行指定的IAM角色具有直接访问 S3 的额外权限，那么即使用户没有 S3 访问权限授予的权限，他们也将能够访问该角色允许的数据。

2. 使用 Amazon EMR 版本标签为 6.15.0 或更高版本并spark-defaults按分类启动您的EMR无服务器应用程序，如下例所示。将 *red text* 中的值替换为适合您的使用场景的适当值。

```
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://DOC-EXAMPLE-BUCKET-OUTPUT/
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"
         }
      }]
}'
```

EMR无服务器的 S3 访问权限授予注意事项

有关在EMR无服务器中使用 Amazon S3 访问授权时的重要支持、兼容性和行为信息，请参阅《亚马逊EMR管理指南》EMR中的 Amazon S3 访问权限授予注意事项。

使用记录 Amazon EMR 无服务器API呼叫 AWS CloudTrail

Amazon EMR Serverless 已与 AWS CloudTrail，一项提供用户、角色或用户所执行操作记录的服务 AWS EMR无服务器中的服务。 CloudTrail 将所有EMR无服务器API调用捕获为事件。捕获的调用包括来自EMR无服务器控制台的调用和对EMR无服务器API操作的代码调用。如果您创建了跟踪，则可以将 CloudTrail事件持续传输到 Amazon S3 存储桶，包括EMR无服务器的事件。如果您未配置跟踪，您仍然可以在 CloudTrail 控制台的 “事件历史记录” 中查看最新的事件。使用收集的信息 CloudTrail，您可以确定向 EMR Serverless 发出的请求、发出请求的 IP 地址、谁发出了请求、何时发出请求以及其他详细信息。

要了解更多信息 CloudTrail，请参阅 AWS CloudTrail 用户指南。

EMR中的无服务器信息 CloudTrail

CloudTrail 已在您的设备上启用 AWS 账户 当你创建账户时。在 EMR Serverless 中发生活动时，该活动会与其他 CloudTrail 活动一起记录在事件中 AWS 事件历史记录中的服务事件。您可以查看、搜索和下载最近发生的事件 AWS 账户。 有关更多信息，请参阅使用事件历史记录查看 CloudTrail 事件。

如需持续记录您的事件 AWS 账户（包括EMR无服务器的事件）创建跟踪。跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。默认情况下，当您在控制台中创建跟踪时，该跟踪将应用于所有跟踪 AWS 区域。 该跟踪记录了来自所有区域的事件 AWS 对日志文件进行分区并将日志文件传送到您指定的 Amazon S3 存储桶。此外，您还可以配置其他 AWS 服务，用于进一步分析 CloudTrail 日志中收集的事件数据并据此采取行动。有关更多信息，请参阅下列内容：

- 创建跟踪记录概述
- CloudTrail 支持的服务和集成
- 为以下各项配置亚马逊SNS通知 CloudTrail
- 接收来自多个地区的 CloudTrail 日志文件和接收来自多个账户的 CloudTrail 日志文件

所有EMR无服务器操作都由《无服务器参考》记录 CloudTrail 并记录在《EMR无服务器API参考》中。例如，调用`StartJobRun`和`CancelJobRun`操作会在 CloudTrail 日志文件中生成条目。`CreateApplication`

每个事件或日记账条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用 root 还是 AWS Identity and Access Management (IAM) 用户凭证。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他人提出 AWS 服务。

有关更多信息，请参阅CloudTrail userIdentity 元素。

了解EMR无服务器日志文件条目

跟踪是一种配置，允许将事件作为日志文件传输到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。事件代表来自任何来源的单个请求，包括有关请求的操作、操作的日期和时间、请求参数等的信息。 CloudTrail 日志文件不是公共API调用的有序堆栈跟踪，因此它们不会按任何特定的顺序出现。

以下示例显示了演示该`CreateApplication`操作的 CloudTrail 日志条目。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:admin",
        "arn": "arn:aws:sts::012345678910:assumed-role/Admin/admin",
        "accountId": "012345678910",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::012345678910:role/Admin",
                "accountId": "012345678910",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-06-01T23:46:52Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-06-01T23:49:28Z",
    "eventSource": "emr-serverless.amazonaws.com",
    "eventName": "CreateApplication",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "203.0.113.0",
    "userAgent": "PostmanRuntime/7.26.10",
    "requestParameters": {
        "name": "my-serverless-application",
        "releaseLabel": "emr-6.6",
```

```
        "type": "SPARK",
        "clientToken": "0a1b234c-de56-7890-1234-567890123456"
    },
    "responseElements": {
        "name": "my-serverless-application",
        "applicationId": "1234567890abcdef0",
        "arn": "arn:aws:emr-serverless:us-west-2:555555555555:/
applications/1234567890abcdef0"
    },
    "requestID": "890b8639-e51f-11e7-b038-EXAMPLE",
    "eventID": "874f89fa-70fc-4798-bc00-EXAMPLE",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "012345678910",
    "eventCategory": "Management"
}
```

Amazon EMR Serverless 合规性验证

EMRServerless的安全性和合规性由第三方审计机构作为多项评估的一部分 AWS 合规计划，包括以下内容：

- 系统和组织控制 (SOC)
- 支付卡行业数据安全标准 (PCIDSS)
- 联邦风险和授权管理计划（美联储RAMP）中等
- Health Insurance 流通与责任法案 () HIPAA

AWS 提供了经常更新的列表 AWS 特定合规计划范围内的服务位于 AWS 按合规计划划分的范围内的服务。

您可以使用以下方式下载第三方审计报告 AWS Artifact。 有关更多信息，请参阅中的下载报告 AWS Artic al。

有关 AWS 合规计划，请参阅 AWS 合规计划。

您在使用 EMR Serverless 时的合规责任取决于数据的敏感性、组织的合规目标以及适用的法律和法规。如果你使用 EMR Serverless 必须遵守诸如HIPAA、或美联储RAMP温和派之类的标准PCI， AWS 提供资源来帮助：

- 《安全与合规性快速入门指南》讨论了架构注意事项以及部署以安全性和合规性为重点的基准环境的步骤 AWS.
- AWS 《客户合规指南》可以帮助您从合规角度了解责任共担模式。这些指南总结了保护的最佳实践 AWS 服务 并将指南映射到跨多个框架（包括美国国家标准与技术研究所 (NIST)、支付卡行业安全标准委员会 (PCI) 和国际标准化组织 (ISO)）的安全控制。
- AWS Config 可用于评估您的资源配置对内部实践、行业指南和法规的遵循情况。
- AWS 合规资源是一系列可能适用于您所在行业和所在地区的工作簿和指南。
- AWS Security Hub 为您提供内部安全状态的全面视图 AWS 并帮助您检查自己是否符合安全行业标准和最佳实践。
- AWS Audit Manager— 这个 AWS 服务 帮助您持续审核您的 AWS 用于简化风险管理以及对法规和行业标准的合规性。

Amazon EMR 无服务器中的弹性

这些区域有： AWS 全球基础设施是围绕着建立的 AWS 地区和可用区。 AWS 区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现失效转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 AWS 地区和可用区，请参阅 AWS 全球基础设施。

除了 AWS 全球基础设施，Amazon S EMR erverless 提供与 Amazon S3 的集成，EMRFS以帮助支持您的数据弹性和备份需求。

Amazon EMR Serverless 中的基础设施安全

作为一项托管服务，Amazon EMR 受以下条款的保护 AWS 全球网络安全。有关信息 AWS 安全服务及其方式 AWS 保护基础架构，请参阅 AWS 云安全。设计你的 AWS 使用基础设施安全最佳实践的环境，请参阅安全支柱中的基础设施保护 AWS 架构精良的框架。

你用 AWS 发布了EMR通过网络访问Amazon的API电话。客户端必须支持以下内容：

- 传输层安全 (TLS)。我们需要 TLS 1.2，建议使用 TLS 1.3。
- 具有完美前向保密性的密码套件 ()，例如（Ephemeral Diffie-HellmanPFS）或（Elliptic C DHE urve Ephemeral Diffie-Hellman）。ECDHE大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与IAM委托人关联的私有访问密钥对请求进行签名。或者你可以使用 AWS Security Token Service (AWS STS) 生成用于签署请求的临时安全证书。

Amazon EMR Serverless 中的配置和漏洞分析

AWS 处理基本的安全任务，例如客户机操作系统 (OS) 和数据库修补、防火墙配置和灾难恢复。这些流程已通过相应第三方审核和认证。有关更多详细信息，请参阅以下资源：

- Amazon EMR Serverless 合规性验证
- 责任共担模式
- Amazon Web Services：安全过程概述（白皮书）

的终端节点和配额 EMR Serverless

服务端点

以编程方式连接到 AWS 服务，则使用终端节点。端点是入口点URL的终端节点 AWS 网络服务。除了标准 AWS 端点，一些 AWS 服务 在选定地区提供FIPS终端节点。下表列出了EMR无服务器的服务端点。有关更多信息，请参阅 AWS 服务 端点。

区域名称	区域	端点	协议
美国东部（俄亥俄州）	us-east-2 （仅限于以下可用区：use2-az1use2-az2、和use2-az3）	emr-serverless.us-east-2.amazonaws.com	HTTPS
美国东部（弗吉尼亚州北部）	us-east-1 （仅限于以下可用区：use1-az1use1-az2、use1-az4、use1-az5、和use1-az6）	emr-serverless.us-east-1.amazonaws.com emr-serverless-fips.us-east-1.amazonaws.com	HTTPS
美国西部（加利福尼亚北部）	us-west-1	emr-serverless.us-west-1.amazonaws.com	HTTPS
美国西部（俄勒冈州）	us-west-2	emr-serverless.us-west-2.amazonaws.com emr-serverless-fips.us-west	HTTPS

区域名称	区域	端点	协议
		-2.amazonaws.com	
非洲（开普敦）	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
亚太地区（雅加达）	ap-southeast-3	emr-serverless.ap-southeast-3.amazonaws.com	HTTPS
亚太地区（孟买）	ap-south-1	emr-serverless.ap-south-1.amazonaws.com	HTTPS
亚太地区（大阪）	ap-northeast-3	emr-serverless.ap-northeast-3.amazonaws.com	HTTPS
亚太地区（首尔）	ap-northeast-2	emr-serverless.ap-northeast-2.amazonaws.com	HTTPS

区域名称	区域	端点	协议
亚太地区（新加坡）	ap-southeast-1	emr-serverless.ap-southeast-1.amazonaws.com	HTTPS
亚太地区（悉尼）	ap-southeast-2	emr-serverless.ap-southeast-2.amazonaws.com	HTTPS
亚太地区（东京）	ap-northeast-1	emr-serverless.ap-northeast-1.amazonaws.com	HTTPS
加拿大（中部）	ca-central-1 （仅限于以下可用区：cac1-az1和cac1-az2）	emr-serverless.ca-central-1.amazonaws.com	HTTPS
欧洲地区（法兰克福）	eu-central-1	emr-serverless.eu-central-1.amazonaws.com	HTTPS
欧洲地区（爱尔兰）	eu-west-1	emr-serverless.eu-west-1.amazonaws.com	HTTPS

区域名称	区域	端点	协议
欧洲地区（伦敦）	eu-west-2	emr-serverless.eu-west-2.amazonaws.com	HTTPS
欧洲地区（米兰）	eu-south-1	emr-serverless.eu-south-1.amazonaws.com	HTTPS
欧洲地区（巴黎）	eu-west-3	emr-serverless.eu-west-3.amazonaws.com	HTTPS
欧洲（西班牙）	eu-south-2	emr-serverless.eu-south-2.amazonaws.com	HTTPS
欧洲（斯德哥尔摩）	eu-north-1	emr-serverless.eu-north-1.amazonaws.com	HTTPS
中东（巴林）	me-south-1	emr-serverless.me-south-1.amazonaws.com	HTTPS
中东 (UAE)	me-central-1	emr-serverless.me-central-1.amazonaws.com	HTTPS

区域名称	区域	端点	协议
南美洲（圣保罗）	sa-east-1	emr-serverless.sa-east-1.amazonaws.com	HTTPS
AWS GovCloud（美国东部）	us-gov-east-1	emr-serverless.us-gov-east-1.amazonaws.com	HTTPS
AWS GovCloud（美国西部）	us-gov-west-1	emr-serverless.us-gov-west-1.amazonaws.com	HTTPS

服务限额

服务配额，也称为限制，是您的服务资源或操作的最大数量 AWS 账户 可以使用。EMRServerless 每分钟收集一次服务配额使用情况指标，并在AWS/Usage命名空间中发布这些指标。

> **Note**
>
> New AWS 账户的初始配额可能较低，但随着时间的推移可能会增加。Amazon EMR Serverless 会监控每个账户内的账户使用情况 AWS 区域，然后根据您的使用量自动增加配额。

下表列出了EMR无服务器的服务配额。有关更多信息，请参阅 AWS 服务 配额。

名称	默认限制	是否可调整？	描述
vCPUs 每个账户的最大并发数	16	是	当前账户 vCPUs 可以同时运行的最大数量 AWS 区域.

API限制

以下内容描述了您的每个区域的API限制 AWS 账户.

资源	默认限额
ListApplications	每秒 10 笔交易。每秒 50 笔交易的爆发。
CreateApplication	每秒 1 笔交易。每秒 25 笔交易的爆发。
DeleteApplication	每秒 1 笔交易。每秒 25 笔交易的爆发。
GetApplication	每秒 10 笔交易。每秒 50 笔交易的爆发。
UpdateApplication	每秒 1 笔交易。每秒 25 笔交易的爆发。
ListJobRuns	每秒 1 笔交易。每秒 25 笔交易的爆发。
StartJobRun	每秒 1 笔交易。每秒 25 笔交易的爆发。
GetDashboardForJobRun	每秒 1 笔交易。每秒 2 笔交易的爆发。
CancelJobRun	每秒 1 笔交易。每秒 25 笔交易的爆发。
GetJobRun	每秒 10 笔交易。每秒 50 笔交易的爆发。
StartApplication	每秒 1 笔交易。每秒 25 笔交易的爆发。
StopApplication	每秒 1 笔交易。每秒 25 笔交易的爆发。

其他考虑因素

以下列表包含EMR无服务器的其他注意事项。

- EMR无服务器在以下版本中可用 AWS 区域:
  - 美国东部（俄亥俄）
  - 美国东部（弗吉尼亚州北部）
  - 美国西部（加利福尼亚北部）
  - 美国西部（俄勒冈州）
  - 非洲（开普敦）
  - Asia Pacific (Hong Kong)
  - 亚太地区（雅加达）
  - 亚太地区（孟买）
  - 亚太地区（大阪）
  - 亚太地区（首尔）
  - 亚太地区（新加坡）
  - 亚太地区（悉尼）
  - 亚太地区（东京）
  - 加拿大（中部）
  - 欧洲地区（法兰克福）
  - 欧洲地区（爱尔兰）
  - 欧洲地区（伦敦）
  - 欧洲地区（米兰）
  - 欧洲地区（巴黎）
  - 欧洲（西班牙）
  - 欧洲（斯德哥尔摩）
  - 中东（巴林）
  - 中东 (UAE)
  - 南美洲（圣保罗）
  - AWS GovCloud（美国东部）
  - AWS GovCloud （美国西部）

有关与这些区域关联的终端节点的列表，请参阅服务端点。

- 作业运行的默认超时时间为 12 小时。你可以使用`startJobRun`API或中的`executionTimeoutMinutes`属性来更改此设置 AWS SDK。如果您希望作业运行永远不会超时，则可以将其设置`executionTimeoutMinutes`为 0。例如，如果您有流式处理应用程序，则可以`executionTimeoutMinutes`将其设置为 0 以允许流式处理作业持续运行。
- 中的`billedResourceUtilization`属性`getJobRun`API显示了聚合 v CPU、内存和存储 AWS 已为任务运行计费。计费资源包括工作人员最低 1 分钟的使用量，以及每位工作人员超过 20 GB 的额外存储空间。这些资源不包括闲置的预初始化工作线程的使用情况。
- 如果没有VPC连接，作业可以访问一些 AWS 服务 端点相同 AWS 区域。 这些服务包括亚马逊 S3、 AWS Glue、Amazon CloudWatch Logs、 AWS KMS, AWS Security Token Service, Amazon DynamoDB，以及 AWS Secrets Manager。 您可以启用VPC连接以访问其他 AWS 服务 通过 AWS PrivateLink，但你不需要这样做。要访问外部服务，您可以使用创建应用程序VPC。
- EMR无服务器不支持HDFS。工作线程上的本地磁盘是临时存储，EMRServerless 在作业运行期间使用它来清理和处理数据。
- EMR无服务器不支持现有emr-dynamodb-connector的。

Amazon EMR 无服务器发布版本

Amazon EMR 版本是一组来自大数据生态系统的开源应用程序。每个版本都包括您在运行任务时选择让 Amazon EMR Serverless 部署和配置的大数据应用程序、组件和功能。

在 Amazon EMR 6.0 及更高版本中，您可以部署EMR无服务器。此部署选项不适用于早期的 Amazon EMR 发行版本。提交任务时，必须指定以下支持的版本之一。

主题

- EMR Serverless 7.2.0
- EMR Serverless 7.1.0
- EMR Serverless 7.0.0
- EMR Serverless 6.15.0
- EMR Serverless 6.14.0
- EMR Serverless 6.13.0
- EMR Serverless 6.12.0
- EMR Serverless 6.11.0
- EMR Serverless 6.10.0
- EMR Serverless 6.9.0
- EMR Serverless 6.8.0
- EMR Serverless 6.7.0
- EMR Serverless 6.6.0

EMR Serverless 7.2.0

下表列出了可用的应用程序版本 EMR Serverless 7.2.0。

应用程序	版本
Apache Spark	3.5.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR无服务器 7.2.0 版本说明

- 带EMR无服务器的 Lak e Formation — 你现在可以使用 AWS Lake Formation 对 S3 支持的数据目录表应用精细的访问控制。此功能允许您为 EMR Serverless Spark 作业中的读取查询配置表、行、列和单元级别的访问控制。有关更多信息，请参阅the section called “Lake Formation FGAC” 和the section called “注意事项”。

EMR Serverless 7.1.0

下表列出了可用的应用程序版本 EMR Serverless 7.1.0。

应用程序	版本
Apache Spark	3.5.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.0.0

下表列出了可用的应用程序版本 EMR Serverless 7.0.0。

应用程序	版本
Apache Spark	3.5.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.15.0

下表列出了可用的应用程序版本 EMR Serverless 6.15.0。

应用程序	版本
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR无服务器 6.15.0 版本说明

- TLS支持 — 在 Amazon EMR Serverless 版本 6.15.0 及更高版本中，您可以在 Spark 作业运行中启用工作人员之间的相互TLS加密通信。启用后，EMRServerless会自动为每个工作人员生成一个唯一的证书，该证书将在作业运行中预置，工作人员在TLS握手期间使用该证书来相互进行身份验证并建立加密通道来安全地处理数据。有关相互TLS加密的更多信息，请参阅工作器间加密。

EMR Serverless 6.14.0

下表列出了可用的应用程序版本 EMR Serverless 6.14.0。

应用程序	版本
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.13.0

下表列出了可用的应用程序版本 EMR Serverless 6.13.0。

应用程序	版本
Apache Spark	3.4.1
Apache Hive	3.1.3

应用程序	版本
Apache Tez	0.10.2

EMR Serverless 6.12.0

下表列出了可用的应用程序版本 EMR Serverless 6.12.0。

应用程序	版本
Apache Spark	3.4.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.11.0

下表列出了可用的应用程序版本 EMR Serverless 6.11.0。

应用程序	版本
Apache Spark	3.3.2
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR无服务器 6.11.0 版本说明

- 访问其他账户中的 S3 资源-在 6.11.0 及更高版本中，您可以配置多个IAM角色，以便在访问不同的 Amazon S3 存储桶时代入 AWS 来自EMR无服务器的账户。

EMR Serverless 6.10.0

下表列出了可用的应用程序版本 EMR Serverless 6.10.0。

应用程序	版本
Apache Spark	3.3.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR无服务器 6.10.0 版本说明

- 对于版本为 6.10.0 或更高版本的EMR无服务器应用程序，该属性的默认值为。`spark.dynamicAllocation.maxExecutors` `infinity`早期版本默认为`100`。有关更多信息，请参阅 Spark 作业属性。

EMR Serverless 6.9.0

下表列出了可用的应用程序版本 EMR Serverless 6.9.0。

应用程序	版本
Apache Spark	3.3.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR无服务器 6.9.0 版本说明

- 适用于 Apache Spark 的 Amazon Redshift 集成包含在亚马逊 6.9.0 及更高EMR版本中。本地集成之前是一种开源工具，现在是 Spark 连接器，您可以将其用于构建 Apache Spark 应用程序，这些应用程序可在 Amazon Redshift 和 Amazon Redshift Serverless 中读取和写入数据。有关更多信息，请参阅 在亚马逊无服务器上使用 Apache Spark 的 Amazon Redshift 集成 EMR。
- EMR无服务器版本 6.9.0 增加了对以下内容的支持 AWS Graviton2 (arm64) 架构。您可以使用`create-application`和的`architecture`参数`update-application`APIs来选择 arm64 架构。有关更多信息，请参阅 Amazon EMR 无服务器架构选项。

- 现在，您可以直接从EMR您的无服务器 Spark 和 Hive 应用程序中导出、导入、查询和加入 Amazon DynamoDB 表。有关更多信息，请参阅 使用 Amazon Serverless 连接到 DynamoDB EMR。

已知问题

- 如果您使用适用于 Apache Spark 的 Amazon Redshift 集成，并且具有 Parquet 格式的时间、timetz、时间戳或 timestamptz（精度为微秒），连接器会将时间值舍入为最接近的毫秒值。解决方法是使用文本卸载格式 `unload_s3_format` 参数。

EMR Serverless 6.8.0

下表列出了可用的应用程序版本 EMR Serverless 6.8.0。

应用程序	版本
Apache Spark	3.3.0
Apache Hive	3.1.3
Apache Tez	0.9.2

EMR Serverless 6.7.0

下表列出了可用的应用程序版本 EMR Serverless 6.7.0。

应用程序	版本
Apache Spark	3.2.1
Apache Hive	3.1.3
Apache Tez	0.9.2

特定于引擎的更改、增强和已解决的问题

下表列出了引擎专用的新功能。

更改	描述
功能	Tez 调度器现在支持抢占 Tez 任务，而不是抢占容器

EMR Serverless 6.6.0

下表列出了可用的应用程序版本 EMR Serverless 6.6.0。

应用程序	版本
Apache Spark	3.2.0
Apache Hive	3.1.2
Apache Tez	0.9.2

EMR无服务器初始发行说明

- EMR无服务器支持 Spark 配置分类`spark-defaults`。这种分类会更改 Spark `spark-defaults.conf` XML 文件中的值。配置分类允许您自定义应用程序。有关更多信息，请参阅配置应用程序。
- EMR无服务器支持 Hive 配置分类`hive-site`、`tez-site`和`emrfs-site`。`core-site`这种分类可以分别更改 Hive 的`hive-site.xml`文件、Tez 的`tez-site.xml`文件、Amazon EMR 的 EMRFS设置或 Hadoop 文件中的值。`core-site.xml`配置分类允许您自定义应用程序。有关更多信息，请参阅配置应用程序。

特定于引擎的更改、增强和已解决的问题

- 下表列出了 Hive 和 Tez 的向后移植。

Hive 和 Tez 发生了变化

更改	描述
逆向移植	TEZ-4430：修复了属性问题 `tez.task.launch.cmd-opts`
逆向移植	HIVE-25971：修复了由于打开缓存线程池而导致的 Tez 任务关闭延迟

文档历史记录

下表描述了自上次发布 EMR Serverless 以来对文档所做的重要更改。有关本文档更新的更多信息，您可以订阅 RSS Feed。

变更	说明	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
更新现有政策。	在 A mazonEMRServerless ServiceRolePolicy 策略中`EC2PolicyStatement` 添加了新的`SidCloudWatchPolicyStatement` 和。	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	配置多个IAM角色，让您在使用与EMR无服务器不同的账户（6.11 及更高版本）中访问 Amazon S3 存储桶时扮演的角色	2023 年 10 月 18 日
新版本	EMR Serverless 6.14.0	2023 年 10 月 17 日
新功能	EMR无服务器的默认应用程序配置	2023 年 9 月 25 日
更新为默认的 Hive 属性	更新了`hive.driver.disk` 、`hive.tez.disk.size` `hive.tez.auto.reducer.parallelism` 、和 `tez.group`	2023 年 9 月 12 日

	ing.min-size Hive 作业属性的默认值。	
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
服务相关角色策略更新	更新了AmazonEMR ServerlessServiceR olePolicy SLR角色以在命名空间中发布账户级别的使用情况。"AWS/Usage"	2023 年 4 月 20 日
EMR Serverless 正式上市 (GA)	这是 EMR Serverless 的第一个公开发布。	2022 年 6 月 1 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。