开发人员指南

# AWS Data Pipeline

API 版本 2012-10-29

# AWS Data Pipeline: 开发人员指南

# Table of Contents

# 什么是 AWS Data Pipeline？

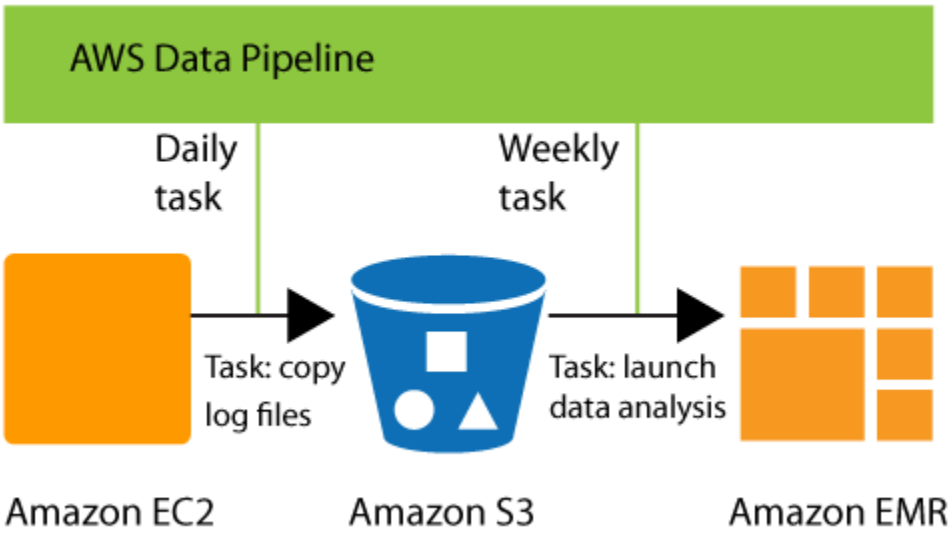
> **Note**
>
> AWS Data Pipeline 服务处于维护模式，没有计划新功能或区域扩展。要了解更多信息并了解如何迁移现有工作负载，请参阅 从 AWS Data Pipeline 迁移工作负载。

AWS Data Pipeline 是一项 Web 服务，您可用于自动处理数据的移动和转换。使用 AWS Data Pipeline，您可以定义数据驱动的工作流，这样任务就可以依赖于前面任务的成功执行。您可以定义数据转换的参数，AWS Data Pipeline 将实施您设置的逻辑。

AWS Data Pipeline 的以下组件协同工作来管理您的数据：

- 管道定义 指定数据管理的业务逻辑。有关更多信息，请参阅管道定义文件语法。
- 管道通过创建 Amazon EC2 实例以执行定义的工作活动，来计划和运行任务。您将管道定义上传到管道，然后激活管道。您可以编辑正在运行的管道的管道定义，并重新激活管道以使其生效。您可以停用管道，修改数据源，然后重新激活管道。完成使用管道后可以将其删除。
- 任务运行程序将轮询任务，然后执行这些任务。例如，任务运行程序可以将日志文件复制到 Amazon S3 并启动 Amazon EMR 集群。任务运行程序已安装，并将在管道定义所创建的资源上自动运行。您可以编写自定义任务运行程序应用程序，也可以使用 AWS Data Pipeline 提供的任务运行程序应用程序。有关更多信息，请参阅任务运行程序。

例如，您每天可使用 AWS Data Pipeline 将 Web 服务器的日志存档到 Amazon Simple Storage Service (Amazon S3)，然后每周对这些日志运行 Amazon EMR (Amazon EMR) 集群以生成流量报告。AWS Data Pipeline 计划每日任务来复制数据，并计划每周任务来启动 Amazon EMR 集群。AWS Data Pipeline 还确保 Amazon EMR 在等待最后一天的数据上传到 Amazon S3 后，再开始其分析，即使存在不可预知的日志上传延迟。




目录



# 从 AWS Data Pipeline 迁移工作负载

AWS 于 2012 年推出了 AWS Data Pipeline 服务。当时，客户正在寻找一种服务，以帮助他们使用各种计算选项在不同的数据源之间可靠地移动数据。现在还有其他服务可以为客户提供更好的体验。例如，您可以使用 AWS Glue 来运行和编排 Apache Spark 应用程序，使用 AWS Step Functions 来帮助编排 AWS 服务组件，或者使用 Amazon Managed Workflows for Apache Airflow (Amazon MWAA)来帮助管理 Apache Airflow 的工作流编排。

本主题介绍如何从 AWS Data Pipeline 迁移到备选选项。选项的选择取决于您在 AWS Data Pipeline 的当前工作负载。您可以将 AWS Data Pipeline 的典型用例迁移到 AWS Glue、AWS Step Functions 或 Amazon MWAA。

# 迁移工作负载到 AWS Glue

AWS Glue 是一项无服务器数据集成服务，可让使用分析功能的用户轻松发现、准备、移动和集成来自多个来源的数据。它包括用于编写、运行任务和编排工作流的工具。通过使用 AWS Glue，您可以发现并连接到 70 多个不同的数据来源，并在集中式数据目录中管理您的数据。您可以直观地创建、运行和监控“提取、转换、加载（ETL）”管道，以将数据加载到数据湖中。此外，您可以使用 Amazon Athena、Amazon EMR 和 Amazon Redshift Spectrum 立即搜索和查询已编目数据。

在下列情况下，我们建议将您的 AWS Data Pipeline 工作负载迁移到 AWS Glue：

- 您正在寻找一种支持各种数据来源、创作界面（包括可视化编辑器和笔记本）以及高级数据管理功能（例如数据质量和敏感数据检测）的无服务器数据集成服务。
- 您的工作负载可以迁移到 AWS Glue 工作流、作业（在 Python 或 Apache Spark 中）和爬网程序中（例如，您的现有管道是在 Apache Spark 之上构建的）。
- 您需要一个能够处理数据管道各个方面的单一平台，包括摄取、处理、传输、完整性测试和质量检查。
- 您的现有管道是根据AWS Data Pipeline控制台上的预定义模板创建的，例如将 DynamoDB 表导出到 Amazon S3，而您正在寻找用途相同的模板。
- 您的工作负载不依赖于特定的 Hadoop 生态系统应用程序，例如 Apache Hive。
- 您的工作负载不需要编排本地服务器。

AWS 按小时费率（按秒计费）收取执行爬网程序（发现数据）和 ETL 任务（处理和加载数据）的费用。AWS GlueStudio 是内置的 AWS Glue 资源编排引擎，不收取额外费用。要了解有关定价的更多信息，请参阅 AWS Glue 定价。

# 将工作负载迁移到 AWS Step Functions

AWS Step Functions 是一项无服务器编排服务，可让您为业务关键型应用程序构建工作流。借助 Step Functions，您可以使用可视化编辑器来构建工作流，并直接与 250 多种AWS服务（例如 AWS Lambda、Amazon EMR、DynamoDB 等）的 11,000 多个操作集成。您可以使用 Step Functions 来编排数据处理管道、处理错误以及处理底层AWS服务的节流限制。您可以创建用于处理和发布机器学习模型、编排微服务以及控制AWS服务的工作流，例如创建提取提取、转换、加载（ETL）工作流。您还可以为需要人工交互的应用程序创建长时间运行的自动化工作流。

与AWS Data Pipeline之类似的是，AWSStep Functions 是由AWS提供的完全托管的服务。您无需管理基础架构、补丁工作人员、管理操作系统版本更新或类似内容。

在下列情况下，我们建议将您的AWS Data Pipeline工作负载迁移到AWS Step Functions：

- 您正在寻找一种无服务器、高度可用的工作流编排服务。
- 您正在寻找一种经济实惠的解决方案，该解决方案可以按单个任务执行的粒度收费。
- 您的工作负载正在为多项其他AWS服务（例如 Amazon EMR、Lambda、 AWS Glue 或 DynamoDB）编排任务。
- 您正在寻找一种低代码解决方案，该解决方案带有用于创建工作流的拖放式可视化设计器，并且不需要学习新的编程概念。
- 您正在寻找一种服务，该服务可以与其他 250 多种 AWS 服务集成，涵盖 11,000 多个开箱即用的操作，并允许与自定义的非 AWS 服务和活动集成。

AWS Data Pipeline 和 Step Functions 都使用 JSON 格式来定义工作流。这允许将您的工作流存储在源代码管理中，管理版本，控制访问权限，并使用 CI/CD 实现自动化。Step Functions 使用一种名为 Amazon State Language 的语法，该语法完全基于 JSON，允许在工作流的文本和视觉表现形式之间实现无缝切换。

使用 Step Functions，您可以选择与当前在 AWS Data Pipeline 中使用的同一版本的 Amazon EMR。

要迁移AWS Data Pipeline托管资源上的活动，您可以在 Step Functions 上使用 AWSSDK 服务集成来自动配置和清理资源。

要在本地服务器、用户管理的 EC2 实例或用户管理的 EMR 集群上迁移活动，您可以为该实例安装 SSM 代理。你可以通过 Step Functions 中的 AWS Systems Manager 运行命令来启动该命令。您也可以根据在 Amazon EventBridge 中定义的计划启动状态机。

AWS Step Functions 有两种类型的工作流：标准工作流和快速工作流。对于标准工作流，您需要根据运行应用程序所需的状态转换次数付费。对于快速工作流，您需要根据工作流的请求数量及其持续时间付费。在 AWS Step Functions定价中了解更多有关定价的信息。

## 将工作负载迁移到 Amazon MWAA

Amazon MWAA (Managed Workflows for Apache Airflow) 是一项适用于 Apache Airflow 的托管式编排服务，让您能够更轻松地在云中大规模设置和操作端到端的数据管道。Apache Airflow 是一种开源工具，用于以编程方式编写、安排和监视被称为 “工作流” 的流程和任务序列。借助 Amazon MWAA，您可以使用 Airflow 和 Python 编程语言来创建工作流程，而无需管理底层基础设施即可实现可扩展性、可用性和安全性。Amazon MWAA 会自动扩展其工作流程执行能力以满足您的需求，并与 AWS 安全服务集成，可帮助您快速、安全地访问数据。

与 AWS Data Pipeline 类似的是，Amazon MWAA 是由 AWS 提供的完全托管服务。虽然您需要学习一些特定于这些服务的新概念，但您无需管理基础设施、补丁工作人员、管理操作系统版本更新或类似内容。

在下列情况下，我们建议将您的 AWS Data Pipeline 工作负载迁移到 Amazon MWAA：

- 您正在寻找一种托管、高度可用的服务来编排用 Python 编写的工作流。
- 您想过渡到完全托管、广泛采用的开源技术 Apache Airflow，以实现最大的便携性。
- 您需要一个能够处理数据管道各个方面的单一平台，包括摄取、处理、传输、完整性测试和质量检查。
- 您正在寻找一种专为数据管道编排而设计的服务，该服务具有丰富的用户界面以实现可观察性、针对失败的工作流重新启动、回填和任务重试等功能。
- 您正在寻找一种包含 800 多个预建操作员和传感器的服务，包括服务 AWS 以及非 AWS 服务。

Amazon MWAA 工作流被定义为使用 Python 的有向无环图 (DAG)，因此您也可以将其视为源代码。Airflow 的可扩展 Python 框架使您能够构建与几乎任何技术连接的工作流。它具有丰富的用户界面，用于查看和监控工作流，并且可以轻松地与版本控制系统集成，以自动执行 CI/CD 流程。

使用亚马逊 MWAA，您可以选择当前正在 AWS Data Pipeline 使用的同一版本的 Amazon EMR。

AWS 按您的 Airflow 环境的运行时间以及为提供更多工作器或 Web 服务器容量而进行的任何额外自动扩缩收费。在 Amazon Managed Workflows for Apache Airflow Pricing 中详细了解定价。

## 映射概念

下表包含服务使用的主要概念的映射。它将帮助熟悉 Data Pipeline 的人理解 Step Functions 和 MWAA 术语。

| Data Pipeline | 连接词 | Step Functions | Amazon MWAA |
| --- | --- | --- | --- |
| 管道 | 工作流程 | 工作流程 | 有向无环图 |
| 管道定义 JSON | 工作流定义或基于 Python 的蓝图 | Amazon States Language JSON | 基于 Python |
| 活动 | 作业 | 状态和任务 | 任务（运算符和传感器） |

| Data Pipeline | 连接词 | Step Functions | Amazon MWAA |
| --- | --- | --- | --- |
| 实例 | 任务运行 | 执行 | DAG 运行 |
| Attempts | 重试尝试 | 缓存和重试器 | 重试 |
| 管道计划 | 计划触发器 | EventBridge 调度器 | Cron、时间表、数据感知 |
| 管道表达式和函数 | 蓝图库 | Step Functions 内置函数和AWS Lambda | 可扩展的 Python 框架 |

# 示例

以下各节列出了一些公开示例，您可以参考这些示例来从 AWS Data Pipeline 迁移到各个服务。您可以将它们作为示例，并根据您的用例对其进行更新和测试，从而在各个服务上构建自己的管道。

## AWS Glue 示例

以下列表包含使用 AWS Glue 的最常见 AWS Data Pipeline 用例的示例实现。

- 运行 Spark 任务
- 将数据从 JDBC 复制到 Amazon S3（包括 Amazon Redshift）
- 将数据从 Amazon S3 复制到 JDBC（包括 Amazon Redshift）
- 将数据从 Amazon S3 复制到 DynamoDB
- 将数据移动到 Amazon Redshift 以及从中移动数据
- 跨账户、跨区域访问 DynamoDB 表

## AWS Step Functions 示例

以下列表包含使用 AWS Step Functions 的最常见 AWS Data Pipeline 用例的示例实现。

- 管理 Amazon EMR 任务
- 在 Amazon EMR Serverless 上运行数据处理任务
- 正在运行 Hive/Pig/Hadoop 任务
- 查询大型数据集（Amazon Athena、Amazon S3、AWS Glue）

- 使用 Amazon Redshift 运行 ETL 工作流
- 编排 AWS Glue 爬网程序

有关使用 AWS Step Functions 的其他教程和示例项目。

### Amazon MWAA 示例

以下列表包含使用的 Amazon MWAA 最常见 AWS Data Pipeline 用例的示例实现。

- 运行 Amazon EMR 任务
- 为 Apache Hive 和 Hadoop 创建自定义插件
- 将数据从 Amazon S3 复制到 Redshift
- 在远程 EC2 实例上执 Shell 脚本
- 编排混合（本地）工作流

有关使用 Amazon MWAA 的其他教程和示例项目。

## 相关服务

AWS Data Pipeline 使用以下服务来存储数据。

- Amazon DynamoDB - 提供低成本、高性能的完全托管 NoSQL 数据库。有关更多信息，请参阅 Amazon DynamoDB 开发人员指南。
- Amazon RDS - 提供扩展到大型数据集的完全托管关系数据库。有关更多信息，请参阅 Amazon Relational Database Service 开发人员指南。
- Amazon Redshift - 提供快速、完全托管的 PB 级数据仓库，可以简便且经济高效地分析大量数据。有关更多信息，请参阅 Amazon Redshift 数据库开发人员指南。
- Amazon S3 - 提供安全、持久且高度可扩展的对象存储。有关更多信息，请参阅 Amazon Simple Storage Service 用户指南。

AWS Data Pipeline 使用以下计算服务来转换数据。

- Amazon EC2 - 提供可调节的计算容量 (简单来说，就是 Amazon 数据中心的服务器)，您可以使用它构建和托管您的软件系统。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南。

- Amazon EMR — 使您能够通过使用框架 (例如 Apache Hadoop 或 Apache Spark) 轻松、快速、经济高效地跨 Amazon EMR 服务器分发和处理大量数据。有关更多信息，请参阅 Amazon EMR 开发人员指南。

## 访问 AWS Data Pipeline

可以使用以下任意接口创建、访问和管理管道：

- AWS Management Console 提供您可用来访问 AWS Data Pipeline 的 Web 界面。
- AWS Command Line Interface (AWS CLI)提供了适用于大量 Amazon Web Services（包括 AWS Data Pipeline）的命令，并在 Windows、macOS 和 Linux 上受支持。有关安装 AWS CLI 的更多信息，请参阅 AWS Command Line Interface。有关 AWS Data Pipeline 命令的列表，请参阅 datapipeline。
- AWS 开发工具包 — 提供特定于语言的 API，并关注许多连接详细信息，例如计算签名、处理请求重试和错误处理。有关更多信息，请参阅 AWS 软件开发工具包。
- 查询 API 提供了您使用 HTTPS 请求调用的低级别 API。使用查询 API 是用于访问 AWS Data Pipeline 的最直接的方式，但需要您的应用程序处理低级别的详细信息，例如生成哈希值以签署请求以及进行错误处理。有关详细信息，请参阅 AWS Data Pipeline API 参考。

## 定价

对于 Amazon Web Services，您只需按实际用量付费。对于 AWS Data Pipeline，您按照活动频率、计划运行的先决条件和位置为管道付费。有关更多信息，请参阅 AWS Data Pipeline 定价。

如果您的 Amazon Web Services account 不到 12 个月，您就有资格使用免费套餐。免费套餐包括每月免费 3 个低频率先决条件和 5 个低频率活动。有关更多信息，请参阅 AWS 免费套餐。

## 管道工作活动支持的实例类型

在 运行管道时，它编译管道组件以创建一组可操作的 Amazon EC2 实例。每个实例包含用于执行特定任务的所有信息。完整的实例集是管道的待办事项列表。AWS Data Pipeline 将实例递交给任务运行程序来处理。

EC2 实例附带了不同的配置，称作实例类型。每个实例类型均有不同的 CPU、输入/输出和存储容量。除了为活动指定实例类型以外，您还可以选择不同的购买选项。并非所有实例类型在所有 Amazon Web Services Region 中都可用。如果实例类型不可用，则您的管道可能无法预置或预置失败。有关实例可用性的信息，请参阅 Amazon EC2 定价页。打开您的实例购买选项的链接，并按区域筛选以查看

某种实例类型在区域中是否可用。有关这些实例类型、系列和虚拟化类型的更多信息，请参阅 Amazon EC2 实例和 Amazon Linux AMI 实例类型矩阵。

下表描述了 AWS Data Pipeline 支持的实例类型。您可以使用 AWS Data Pipeline 在任何区域中启动 Amazon EC2 实例，包括不支持 AWS Data Pipeline 的区域。有关支持 AWS Data Pipeline 的区域的信息，请参阅 Amazon Web Services Region 和终端节点。

目录



## Amazon Web Services Region 的默认 Amazon EC2 实例

如果未在管道定义中指定实例类型，则 AWS Data Pipeline 默认启动一个实例。

下表列出了 AWS Data Pipeline 在支持 AWS Data Pipeline 的区域中默认使用的 Amazon EC2 实例。

| 区域名称 | 区域 | 实例类型 |
| --- | --- | --- |
| 美国东部（弗吉尼亚州北部） | us-east-1 | m1.small |
| 美国西部（俄勒冈州） | us-west-2 | m1.small |
| Asia Pacific (Sydney) | ap-southeast-2 | m1.small |
| 亚太地区（东京) | ap-northeast-1 | m1.small |
| 欧洲 (爱尔兰) | eu-west-1 | m1.small |

下表列出了 AWS Data Pipeline 在不支持 AWS Data Pipeline 的区域中默认启动的 Amazon EC2 实例。

| 区域名称 | 区域 | 实例类型 |
| --- | --- | --- |
| US East (Ohio) | us-east-2 | t2.small |
| 美国西部（北加利福尼亚） | us-west-1 | m1.small |

| 区域名称 | 区域 | 实例类型 |
| --- | --- | --- |
| 亚太地区（孟买） | ap-south-1 | t2.small |
| 亚太地区（新加坡） | ap-southeast-1 | m1.small |
| 亚太地区（首尔） | ap-northeast-2 | t2.small |
| Canada (Central) | ca-central-1 | t2.small |
| 欧洲 (法兰克福) | eu-central-1 | t2.small |
| 欧洲 (伦敦) | eu-west-2 | t2.small |
| 欧洲 (巴黎) | eu-west-3 | t2.small |
| South America (São Paulo) | sa-east-1 | m1.small |

## 支持的额外 Amazon EC2 实例

除了在管道定义中未指定实例类型时创建的默认实例以外，还支持以下实例。

下表列出了 AWS Data Pipeline 支持并可以创建的 Amazon EC2 实例（如果指定）。

| 实例类 | 实例类型 |
| --- | --- |
| 通用型 | t2.nano \| t2.micro \| t2.small \| t2.medium \| t2.large |
| 计算优化 | c3.large \| c3.xlarge \| c3.2xlarge \| c3.4xlarge \| c3.8xlarge \| c4.large \| c4.xlarge \| c4.2xlarge \| c4.4xlarge \| c4.8xlarge \| c5.xlarge \| c5.9xlarge \| c5.2xlarge \| c5.4xlarge \| c5.9xlarge \| c5.18xlarge \| c5d.xlarge \| c5d.2xlarge \| c5d.4xlarge \| c5d.9xlarge \| c5d.18xlarge |
| 内存优化 | m3.medium \| m3.large \| m3.xlarge \| m3.2xlarge \| m4.large \| m4.xlarge \| m4.2xlarge \| m4.4xlarge \| m4.10xlarge \| m4.16xlarge \| m5.xlarge \| m5.2xlarge \| m5.4xlarge \| m5.12xlarge \| m5.24xlarge \| m5d.xlarge \| m5d.2xlarge \| m5d.4xlarge \| m5d.12xlarge \| m5d.24xlarge |

| 实例类 | 实例类型 |
| --- | --- |
| | r3.large \| r3.xlarge \| r3.2xlarge \| r3.4xlarge \| r3.8xlarge \| r4.large \| r4.xlarge \| r4.2xlarge \| r4.4xlarge \| r4.8xlarge \| r4.16xlarge |
| 存储优化 | i2.xlarge \| i2.2xlarge \| i2.4xlarge \| i2.8xlarge \| hs1.8xlarge \| g2.2xlarge \| g2.8xlarge \| d2.xlarge \| d2.2xlarge \| d2.4xlarge \| d2.8xlarge |

## Amazon EMR 集群支持的 Amazon EC2 实例

该表列出了 AWS Data Pipeline 支持并可以为 Amazon EMR 集群创建的 Amazon EC2 实例（如果指定）。有关更多信息，请参阅《Amazon EMR 管理指南》中的支持的实例类型。

| 实例类 | 实例类型 |
| --- | --- |
| 通用型 | m1.small \| m1.medium \| m1.large \| m1.xlarge \| m3.xlarge \| m3.2xlarge |
| 计算优化 | c1.medium \| c1.xlarge \| c3.xlarge \| c3.2xlarge \| c3.4xlarge \| c3.8xlarge \| cc1.4xlarge\| cc2.8xlarge \| c4.large \| c4.xlarge \| c4.2xlarge\| c4.4xlarge \| c4.8xlarge \| c5.xlarge \| c5.9xlarge \| c5.2xlarge \| c5.4xlarge \| c5.9xlarge \| c5.18xlarge \| c5d.xlarge \| c5d.2xlarge \| c5d.4xlarge \| c5d.9xlarge \| c5d.18xlarge |
| 内存优化 | m2.xlarge \| m2.2xlarge \| m2.4xlarge \| r3.xlarge \| r3.2xlarge \| r3.4xlarge \| r3.8xlarge \| cr1.8xlarge \| m4.large \| m4.xlarge \| m4.2xlarge \| m4.4xlarge \| m4.10xlarge \| m4.16large \| m5.xlarge \| m5.2xlarge \| m5.4xlarge \| m5.12xlarge \| m5.24xlarge \| m5d.xlarge \| m5d.2xlarge \| m5d.4xlarge \| m5d.12xlarge \| m5d.24xlarge \| r4.large \| r4.xlarge \| r4.2xlarge \| r4.4xlarge \| r4.8xlarge \| r4.16xlarge |
| 存储优化 | h1.4xlarge \| hs1.2xlarge \| hs1.4xlarge\| hs1.8xlarge \| i2.xlarge \| i2.2xlarge \| i2.4large \| i2.8xlarge \| d2.xlarge \| d2.2xlarge\| d2.4xlarge \| d2.8xlarge |
| 加速计算 | g2.2xlarge \| cg1.4xlarge |

# AWS Data Pipeline 概念

在开始之前，请了解有关 AWS Data Pipeline 的主要概念和组件。

目录



## 管道定义

管道定义是指您如何将业务逻辑传达给 AWS Data Pipeline。它包含以下信息：

- 您数据源的名称、位置和格式
- 转换数据的活动
- 这些活动的计划
- 运行您的活动和先决条件的资源
- 必须满足先决条件，然后才能计划活动
- 在管道执行继续时提醒您状态更新的方式

AWS Data Pipeline 根据您的管道定义确定任务、计划任务并将任务分配给任务运行程序。如果任务未成功完成，AWS Data Pipeline 会根据您的指令重试任务，如有必要，将任务重新分配给其他任务运行程序。如果任务反复失败，您可以配置管道通知您。

例如，您可以在管道定义中指定，您的应用程序在 2013 年的每个月生成的日志文件将存档于 Amazon S3 存储桶。然后 AWS Data Pipeline 将创建 12 个任务，每个任务复制一个月的数据，不论该月有 30、31、28 还是 29 天。

您可以通过下列方法之一来创建管道定义：

- 使用 AWS Data Pipeline 控制台通过图形方式创建
- 以文本方式，按命令行界面使用的格式编写 JSON 文件
- 使用 AWS 开发工具包或 AWS Data Pipeline API 以编程方式调用 Web 服务

管道定义可以包含以下类型的组件。

管道组件

数据节点

任务的输入数据的位置，或者存储输出数据的位置。

活动

按计划执行的工作的定义，使用计算资源，通常有输入和输出数据节点。

先决条件

必须为 true 然后操作才能运行的条件语句。

资源

执行管道定义的工作的计算资源。

操作

在满足指定条件时触发的操作，如活动故障。

有关更多信息，请参阅管道定义文件语法。

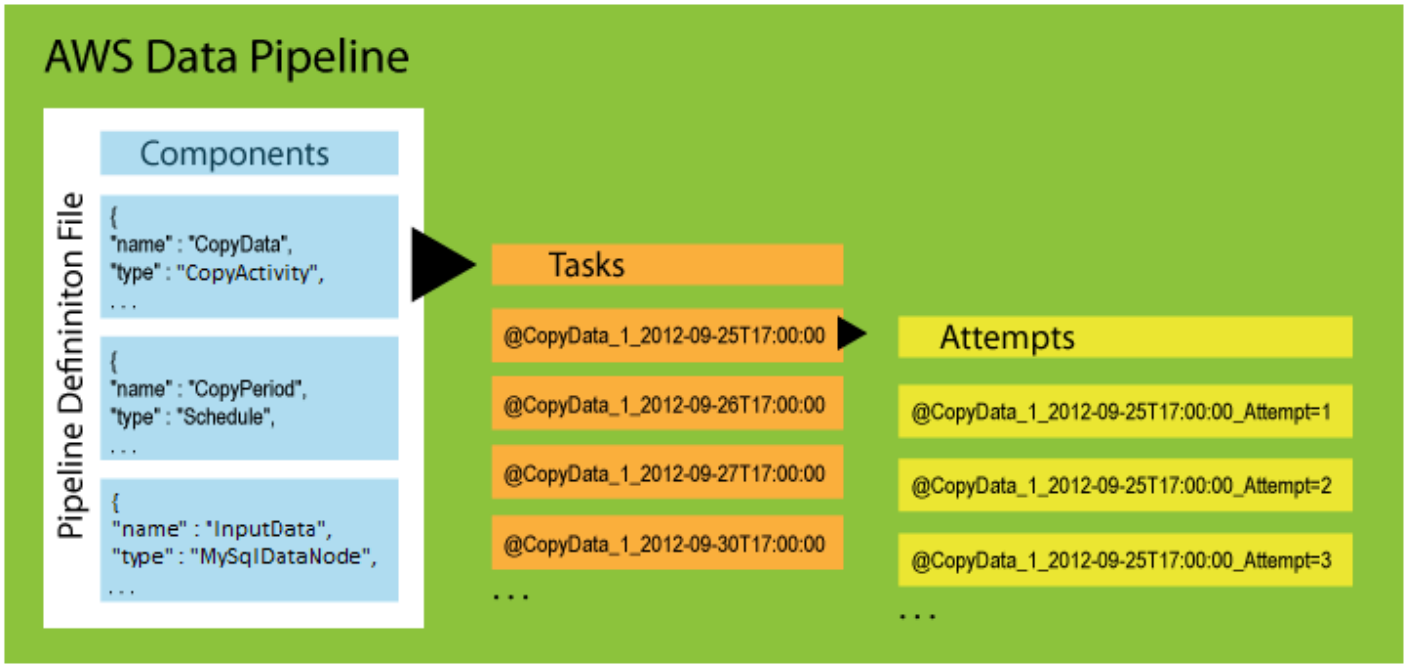
# 管道组件、实例和尝试

有三种类型的项与计划管道关联：

- 管道组件 - 管道组件提供管道的业务逻辑，由管道定义的不同部分来表示。管道组件指定数据源、活动、计划和工作流的先决条件。它们可以从父组件继承属性。组件之间的关系由引用来定义。管道组件定义数据管理规则。
- 实例 - 当 AWS Data Pipeline 运行管道时，它编译管道组件以创建一组可操作的实例。每个实例包含用于执行特定任务的所有信息。完整的实例集是管道的待办事项列表。AWS Data Pipeline 将实例递交给任务运行程序来处理。

- 尝试 - 为了提供稳定的数据管理，AWS Data Pipeline 会重试失败的操作。它会继续执行，直到任务达到允许的最大重试次数。尝试对象跟踪各种尝试、结果和故障原因 (如果适用)。实质上，它是一个带有计数器的实例。AWS Data Pipeline 使用与上一次尝试相同的资源来执行重试，例如 Amazon EMR 集群和 EC2 实例。

> **Note**
>
> 重试失败的任务是容错战略的一个重要组成部分，AWS Data Pipeline 定义提供条件和阈值来控制重试。但是，重试次数太多可能会延迟不可恢复故障的检测，因为 AWS Data Pipeline 在用尽您指定的全部重试次数之前不报告故障。如果运行在 AWS 资源上，过多的重试可能会产生额外的费用。因此，对于您用来控制重试次数的 AWS Data Pipeline 的默认设置以及相关设置，请仔细考虑什么时候才适合超过该设置。

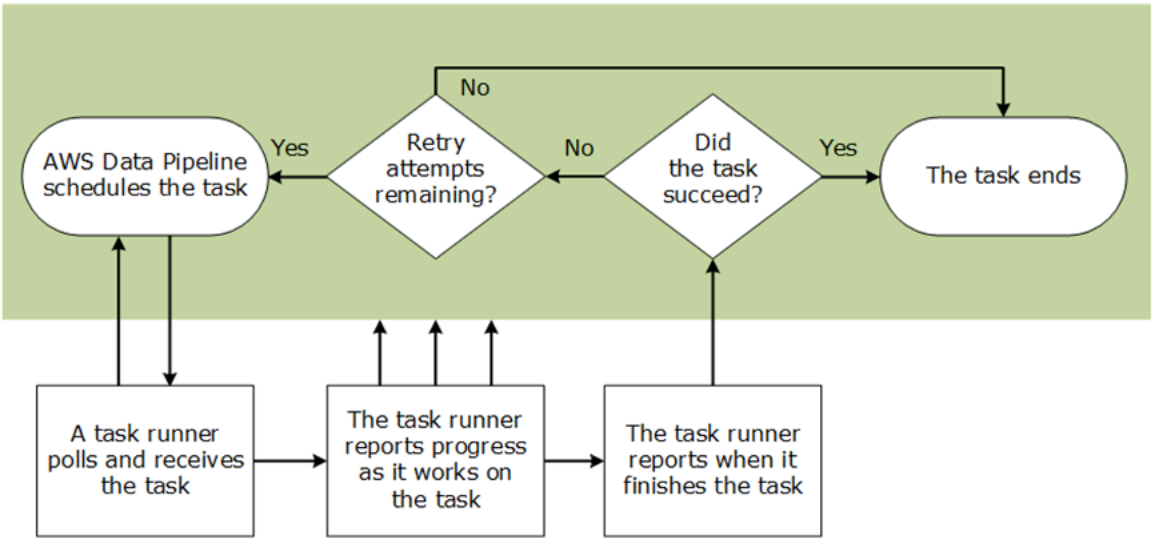



# 任务运行程序

任务运行程序是轮询 AWS Data Pipeline 以获取任务然后执行这些任务的应用程序。

任务运行程序是由 AWS Data Pipeline 提供的任务运行程序的默认实施。安装和配置了任务运行程序时，它将轮询 AWS Data Pipeline 以获取与您所激活管道关联的任务。将某个任务分配给任务运行程序时，它会执行该任务并将状态报告回 AWS Data Pipeline。

下图介绍了 AWS Data Pipeline 如何与任务运行程序交互来处理计划的任务。任务是 AWS Data Pipeline 服务与任务运行程序共享的独立任务单位。它与管道不同，后者是通常会产生多个任务的活动和资源的一般定义。




您可以通过以下两种方式使用任务运行程序来处理您的管道：

- AWS Data Pipeline 在由 AWS Data Pipeline Web 服务启动和管理的资源上为您安装任务运行程序。
- 您可以在自己管理的计算资源上安装任务运行程序，例如长时间运行的 EC2 实例或本地服务器。

有关使用任务运行程序的更多信息，请参阅使用任务运行程序。

# 数据节点

在 AWS Data Pipeline 中，数据节点定义管道活动用作输入或输出的位置和数据类型。AWS Data Pipeline 支持以下类型的数据节点：

DynamoDBDataNode

包含 HiveActivity 或 EmrActivity 要使用数据的 DynamoDB 表。

SqlDataNode

表示管道活动要使用的数据的 SQL 表和数据库查询。

> **Note**
>
> 以前使用 MySqlDataNode。现在改用 SqlDataNode。

RedshiftDataNode

包含 RedshiftCopyActivity 要使用数据的 Amazon Redshift 表。

S3DataNode

包含供管道活动使用的一个或多个文件的 Amazon S3 位置。

## 数据库

AWS Data Pipeline 支持以下类型的数据库：

JdbcDatabase

JDBC 数据库。

RdsDatabase

Amazon RDS 数据库。

RedshiftDatabase

Amazon Redshift 数据库。

## 活动

在 AWS Data Pipeline 中，活动是定义要执行的工作的管道组件。AWS Data Pipeline 提供了几种适用于常见场景的预打包活动，例如将数据从一个位置移动到另一个位置，运行 Hive 查询等。活动是可扩展的，因此您可以运行自己的自定义脚本以支持随意组合。

AWS Data Pipeline 支持以下类型的活动：

CopyActivity

将数据从一个位置复制到另一个位置。

EmrActivity

运行 Amazon EMR 集群。

HiveActivity

在 Amazon EMR 集群上运行 Hive 查询。

HiveCopyActivity

在 Amazon EMR 集群上运行 Hive 查询，支持高级数据筛选并支持 S3DataNode 和 DynamoDBDataNode。

PigActivity

在 Amazon EMR 集群上运行 Pig 脚本。

RedshiftCopyActivity

向 Amazon Redshift 表复制数据和从其中复制数据。

ShellCommandActivity

以活动的方式运行自定义 UNIX/Linux shell 命令。

SqlActivity

对数据库运行 SQL 查询。

一些活动提供特殊的暂存数据和数据库表支持。有关更多信息，请参阅将数据和表与管道活动一起暂存。

# 先决条件

在 AWS Data Pipeline 中，先决条件是管道组件，其中包含的条件语句必须为 true，然后活动才能运行。例如，先决条件可以在管道活动尝试复制源数据之前检查源数据是否存在。AWS Data Pipeline 提供了几种适应常见场景的预打包先决条件，如数据库表是否存在，Amazon S3 密钥是否存在等。不过，先决条件是可扩展的，允许您运行自己的自定义脚本以支持随意组合。

有两种类型的先决条件：系统管理的先决条件和用户管理的先决条件。系统管理的先决条件由 AWS Data Pipeline Web 服务代表您运行，不需要计算资源。用户管理的先决条件仅在您使用 `runsOn` 或 `workerGroup` 字段指定的计算资源上运行。`workerGroup` 资源派生自使用先决条件的活动。

## 系统管理的先决条件

DynamoDBDataExists

检查特定 DynamoDB 表中是否存在数据。

DynamoDBTableExists

检查是否存在 DynamoDB 表。

S3KeyExists

检查是否存在 Amazon S3 密钥。

S3PrefixNotEmpty

检查 Amazon S3 前缀是否为空。

## 用户管理的先决条件

存在

检查数据节点是否存在。

ShellCommandPrecondition

运行自定义 Unix/Linux shell 命令作为先决条件。

# 资源

在 AWS Data Pipeline 中，资源是执行管道活动所指定工作的计算资源。AWS Data Pipeline 支持以下类型的资源：

Ec2Resource

执行管道活动定义的工作的 EC2 实例。

EmrCluster

执行管道活动所定义工作的 Amazon EMR 集群，例如 EmrActivity。

资源可以运行在与其工作数据集相同的区域中，即使该区域与 AWS Data Pipeline 不同。有关更多信息，请参阅利用多个区域中的资源使用管道。

## 资源限制

AWS Data Pipeline 扩展以满足大量并发任务，您可以将其配置为自动创建所需的资源以处理大型工作负载。这些自动创建的资源由您控制，并计入您的 Amazon Web Services account 资源限制。例如，如果您配置 AWS Data Pipeline 自动创建 20 个节点的 Amazon EMR 集群以处理数据，并且您的 Amazon Web Services account 的 EC2 实例限制设置为 20，您可能会无意中用尽可用的回填资源。因此，在设计中请考虑这些资源限制或相应增加您的账户限制。有关服务限制的更多信息，请参阅 AWS 一般参考中的 AWS 服务限制。

> **Note**
>
> 每个 `Ec2Resource` 组件对象的限制是一个实例。

## 支持的平台

管道可以在以下平台中启动您的资源：

EC2-Classic

您的资源会在一个可与其他客户共享的扁平化网络中运行。

EC2-VPC

您的资源会在一个逻辑上与 Amazon Web Services account 分离的 Virtual Private Cloud (VPC) 中运行。

根据各地区的不同条件，您的 Amazon Web Services account 可以在两个平台或只能在 EC2-VPC 中启动资源。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的支持的平台。

如果您的 Amazon Web Services account 仅支持 EC2-VPC，我们会在各个 Amazon Web Services Region 中为您创建一个默认 VPC。默认情况下，我们将在您的默认 VPC 的默认子网中启动资源。或者，在配置资源时，您可以创建一个非默认 VPC，并指定其中一个子网，然后在非默认 VPC 的指定子网中启动资源。

在 VPC 中启动实例时，您必须指定一个专门为该 VPC 创建的安全组。在 VPC 中启动实例时，您无法指定为 EC2-Classic 创建的安全组。此外，您必须使用安全组 ID 而非安全组名称来识别 VPC 的安全组。

# Amazon EC2 竞价型实例与 Amazon EMR 集群和 AWS Data Pipeline

管道可以将 Amazon EC2 竞价型实例用于其 Amazon EMR 集群资源中的任务节点。默认情况下，管道使用按需实例。Spot 实例允许您使用的备用 EC2 实例并运行它们。Spot 实例定价模型是对按需定价模型和预留实例定价模型的补充，为用户提供了对于应用需要的计算容量而言可能是最经济实惠的价格选项。有关更多信息，请参阅 Amazon EC2 Spot 实例产品页面。

当您使用竞价型实例时，AWS Data Pipeline 在您启动集群时将竞价型实例最高出价提交给 Amazon EMR。它会自动将集群的工作分配到您使用 `taskInstanceCount` 字段定义的数个 Spot 实例任务节点。AWS Data Pipeline 限制任务节点的 Spot 实例，以确保按需核心节点可用于运行您的管道。

您可以编辑发生故障或已完成的管道资源实例，以添加 Spot 实例。管道重新启动集群时，会将 Spot 实例作为任务节点。

## Spot 实例注意事项

在您将 Spot 实例与 AWS Data Pipeline 结合使用时，应注意以下事项：

- 您的竞价型实例可能因竞价型实例的价格超出您的最高出价或者 Amazon EC2 容量的原因而终止。但是，您不会丢失您的数据，因为 AWS Data Pipeline 所用集群的核心节点始终为按需实例，不受终止影响。
- 当 Spot 实例异步满足容量时，可能需要较长时间才能启动。因此，Spot 实例的管道比同等按需实例管道运行慢。
- 如果您未收到 Spot 实例 (例如当您的最高出价太低时)，您的集群可能不运行。

# 操作

AWS Data Pipeline 操作是管道组件在发生特定事件时执行的步骤，例如成功、失败或延迟活动。活动的事件字段指的是操作，例如对 `EmrActivity` 的 `onLateAction` 字段中 `snsalarm` 的引用。

AWS Data Pipeline 依赖于 Amazon SNS 通知作为在自动模式下指示管道及其组件状态的主要方式。有关更多信息，请参阅 Amazon SNS。除了 SNS 通知之外，您还可以使用 AWS Data Pipeline 控制台和 CLI 来获取管道状态信息。

AWS Data Pipeline 支持以下操作：

SnsAlarm

根据 `onSuccess`、`OnFail` 和 `onLateAction` 事件发送 SNS 通知到主题的操作。

终止

触发对挂起或未完成活动、资源或数据节点进行取消的操作。您不能终止包括 onSuccess、OnFail 或 onLateAction 的操作。

## 主动监控管道

检测问题的最佳方式是从一开始就主动地监控您的管道。您可以配置管道组件告知您特定情况或事件，例如管道组件失败时或者未按计划开始时间启动时。AWS Data Pipeline 在管道组件上提供了可与 Amazon SNS 通知关联的事件字段，例如 onSuccess、OnFail 和 onLateAction，从而简化了通知的配置。

# 对 AWS Data Pipeline 进行设置

首次使用 AWS Data Pipeline 之前，请完成以下任务。

任务

- 注册 AWS
- 为 AWS Data Pipeline 和管道资源创建 IAM 角色
- 允许 IAM 主体（用户和群组）执行必要操作
- 授权以编程方式访问

完成这些任务之后，您可以开始使用 AWS Data Pipeline。有关基本教程，请参阅 开始使用 AWS Data Pipeline。

# 注册 AWS

当您注册 Amazon Web Services (AWS) 时，您的 Amazon Web Services account 会自动注册 AWS 中的所有服务，包括 AWS Data Pipeline。您只需为使用的服务付费。有关 AWS Data Pipeline 使用费率的更多信息，请参阅 AWS Data Pipeline。

## 注册 AWS 账户

如果您还没有 AWS 账户，请完成以下步骤来创建一个。

注册 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
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。

   当您注册 AWS 账户时，系统将会创建一个 AWS 账户根用户。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为管理用户分配管理访问权限，并且只使用根用户执行需要根用户访问权限的任务。

注册过程完成后，AWS 会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

## 创建管理用户

注册 AWS 账户后，请保护好您的 AWS 账户根用户，启用 AWS IAM Identity Center，并创建一个管理用户，以避免使用根用户执行日常任务。

保护您的 AWS 账户根用户

1. 选择根用户并输入您的 AWS 账户电子邮件地址，以账户拥有者身份登录 AWS Management Console。在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的以根用户身份登录。
2. 对您的根用户启用多重身份验证（MFA）。

   有关说明，请参阅《IAM 用户指南》中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建管理用户

1. 启用 IAM Identity Center

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的 启用 AWS IAM Identity Center。
2. 在 IAM Identity Center 中，为管理用户授予管理访问权限。

   有关将 IAM Identity Center 目录 用作身份源的教程，请参阅《AWS IAM Identity Center 用户指南》中的 Configure user access with the default IAM Identity Center 目录。

作为管理用户登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  要获取使用 IAM Identity Center 用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的登录 AWS 访问门户。

# 为 AWS Data Pipeline 和管道资源创建 IAM 角色

AWS Data Pipeline 需要 IAM 角色来确定执行操作和访问 AWS 资源的权限。管道角色决定 AWS Data Pipeline 拥有的权限，资源角色决定在管道资源（例如 EC2 实例）上运行的应用程序所拥有的权限。在您创建管道时，请指定这些角色。即使您未指定自定义角色并使用默认角色

DataPipelineDefaultRole 和 DataPipelineDefaultResourceRole，也必须先创建角色并附加权限策略。有关更多信息，请参阅适用于 AWS Data Pipeline 的 IAM 角色。

## 允许 IAM 主体（用户和群组）执行必要操作

要使用管道，必须允许您账户中的 IAM 主体（用户或群组）对管道定义的其他服务执行所需的 AWS Data Pipeline 操作。

为了简化权限，您可以将 AWSDataPipeline_FullAccess 托管策略附加到 IAM 主体。此托管策略允许主体执行用户需要的所有操作以及未指定自定义角色时使用 AWS Data Pipeline 的默认角色的 iam:PassRole 操作。

我们强烈建议您仔细评估此托管策略，将权限仅限于您的用户所需的权限。如有必要，请使用此策略作为起点，然后移除权限以创建限制性更强的内联权限策略，您可以将其附加到 IAM 主体。有关示例权限策略的更多信息，请参阅 AWS Data Pipeline 策略示例。

与以下示例类似的策略语句必须包含在附加到任何使用管道的 IAM 主体的策略中。此语句允许 IAM 主体对管道使用的角色执行 PassRole 操作。如果您不使用默认角色，请使用您创建的自定义角色替换 *MyPipelineRole* 和 *MyResourceRole*。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "iam:PassRole",
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iam::*:role/MyPipelineRole",
                "arn:aws:iam::*:role/MyResourceRole"
            ]
        }
    ]
}
```

以下过程介绍了如何创建 IAM 组，把 AWSDataPipeline_FullAccess 托管策略附加到组，然后把用户添加到组。您可以将此过程用于任何内联策略

创建用户组 **DataPipelineDevelopers** 并附加 AWSDataPipeline_FullAccess 策略

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。

2. 在导航窗格中，依次选择 Groups (组) 和 Create New Group (创建新组)。
3. 输入一个组的名称（例如 **DataPipelineDevelopers**），然后选择下一步。
4. 在筛选中输入 **AWSDataPipeline_FullAccess**，然后从列表中将其选中。
5. 选择 Next Step，然后选择 Create Group。
6. 要将用户添加到组：
    a. 从组列表中选择您创建的组。
    b. 依次选择 Group Actions (组操作)、Add Users to Group (将多个用户添加到组)。
    c. 从列表中选择要添加的用户，然后选择添加用户到组。

# 授权以编程方式访问

如果用户需要在 AWS Management Console 之外与 AWS 交互，则需要编程式访问权限。授予编程式访问权限的方法取决于访问 AWS 的用户类型。

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<br><br>(在 IAM Identity Center 中管理的用户) | 使用临时凭证签署向 AWS CLI、AWS软件开发工具包或 AWS API 发出的编程请求。 | 按照您希望使用的界面的说明进行操作。<br><br>• 有关 AWS CLI 的更多信息，请参阅《AWS Command Line Interface 用户指南》中的配置 AWS CLI 以使用 AWS IAM Identity Center。<br>• 有关 AWS软件开发工具包、工具和 AWS API 的更多信息，请参阅《AWS软件开发工具包和工具参考指南》中的 IAM Identity Center 身份验证。 |

| 哪个用户需要编程式访问权限？ | 目的 | 方式 |
| --- | --- | --- |
| IAM | 使用临时凭证签署向 AWS CLI、AWS软件开发工具包或 AWS API 发出的编程请求。 | 按照《IAM 用户指南》中将临时凭证用于 AWS 资源中的说明进行操作。 |
| IAM | (不推荐使用)<br>使用长期凭证签署向 AWS CLI、AWS软件开发工具包或 AWS API 发出的编程请求。 | 按照您希望使用的界面的说明进行操作。<br>• 有关 AWS CLI 的更多信息，请参阅《AWS Command Line Interface 用户指南》中的使用 IAM 用户凭证进行身份验证。<br>• 有关 AWS 软件开发工具包和工具的更多信息，请参阅《AWS 软件开发工具包和工具参考指南》中的使用长期凭证进行身份验证。<br>• 有关 AWS API 的更多信息，请参阅《IAM 用户指南》中的管理 IAM 用户的访问密钥。 |

# 开始使用 AWS Data Pipeline

AWS Data Pipeline 可帮助您排列、计划、运行和管理定期数据处理工作负载，可靠且经济实惠。此服务让您可以轻松地根据自己的业务逻辑，在本地和在云中使用结构化和非结构化数据设计提取-转换-加载 (ETL) 活动。

要使用 AWS Data Pipeline，您将创建一个为数据处理指定业务逻辑的管道定义。典型的管道定义由定义要执行工作的活动、定义输入和输出数据的位置和类型的数据节点组成。

在本教程中，您将运行 shell 命令脚本，统计 Apache Web 服务器日志中的 GET 请求数。此管道在 1 小时内每 15 分钟运行一次，并将每次迭代的输出写入 Amazon S3 中。

先决条件

在开始之前，请完成对 AWS Data Pipeline 进行设置中的任务。

管道对象

管道使用以下对象：

ShellCommandActivity

读取输入日志文件并统计错误数。

S3DataNode (input)

包含输入日志文件的 S3 存储桶。

S3DataNode (output)

用于输出的 S3 存储桶。

Ec2Resource

AWS Data Pipeline 执行活动使用的计算资源。

请注意，如果您有大量日志文件数据，则可以配置管道使用 EMR 集群来处理文件，而不是 EC2 实例。

计划

定义活动在 1 小时内每 15 分钟执行一次。

任务

- 创建管道
- 监控正在运行的管道
- 查看输出
- 删除管道

# 创建管道

开始使用 AWS Data Pipeline 最快捷的方式就是使用称为模板 的管道定义。

创建管道

1. 打开 AWS Data Pipeline 控制台，网址为 https://console.aws.amazon.com/datapipeline/。
2. 从导航栏中选择区域。您可以选择向您提供的任何区域，无需理会您身处的位置。许多 AWS 资源特定于某个区域，但 AWS Data Pipeline 使您能够使用与管道位于不同区域中的资源。
3. 您看到的第一个屏幕取决于您是否在当前区域创建了管道。
   a. 如果您尚未在此区域创建管道，则控制台会显示简介屏幕。选择 Get started now。
   b. 如果您已经在此区域创建了管道，则控制台会显示一个页面，其中列出了您在该区域的管道。选择创建新管道。
4. 在名称中，输入管道的名称。
5. （可选）对于描述，输入管道的描述。
6. 对于 Source，选择 Build using a template，然后选择以下模板：Getting Started using ShellCommandActivity。
7. 在您选择模板时打开的 Parameters 部分下，将 S3 input folder 和 Shell command to run 保留为其默认值。单击 S3 output folder 旁边的文件夹图标，选择您的存储桶或文件夹之一，然后单击 Select。
8. 在 Schedule 下，保留默认值。当您激活管道时，管道开始运行，每 15 分钟运行一次，连续运行一小时。

   如果您愿意，您可以改为选择 Run once on pipeline activation。
9. 在管道配置下，将日志记录保持为启用状态。选择日志的 S3 位置下的文件夹图标，选择您的一个存储桶或文件夹，然后选择选择。

   如果您愿意，您也可以禁用日志记录。

10. 在安全/访问下，将 IAM 角色设置为默认。
11. 单击 Activate。

    如果您愿意，您可以选择在 Architect 中编辑来修改此管道。例如，您可以添加先决条件。

# 监控正在运行的管道

在激活管道后，您将转至 Execution details 页面，可在其中监控管道的进度。

监控管道的进度

1. 单击 Update 或按 F5 以更新显示的状态。

   > **Tip**
   >
   > 如果未列出任何运行，请确保 Start (in UTC) 和 End (in UTC) 包含管道的计划开始时间和结束时间，然后单击 Update。

2. 如果管道中的每个对象的状态均为 `FINISHED`，则表示管道已成功完成计划的任务。
3. 如果您的管道未成功完成，请检查您的管道设置是否有问题。有关管道的实例运行失败或未完成的问题排查的更多信息，请参阅解决常见问题。

# 查看输出

打开 Amazon S3 控制台并导航到您的存储桶。如果您在 1 小时中每 15 分钟运行一次管道，则会看到带有时间戳的子文件夹。每个子文件夹中包含一个名为 `output.txt` 的文件。由于我们每次在同一个输入文件上运行脚本，输出文件相同。

# 删除管道

要停止产生费用，请删除您的管道。删除管道会删除管道定义和所有关联对象。

删除管道

1. 在列出管道页面中选择管道。
2. 单击操作，然后选择删除。
3. 当系统提示进行确认时，选择 Delete（删除）。

如果您完成了本教程的输出，请从您的 Amazon S3 存储桶删除输出文件夹。

# 使用管道

您可以使用命令行界面 (CLI) 或 AWS 软件开发工具包来管理、创建和修改管道。以下章节介绍了基本 AWS Data Pipeline 概念并向您演示如何使用管道。

> **Important**
>
> 在开始之前，请参阅对 AWS Data Pipeline 进行设置。

目录



## 创建管道

AWS Data Pipeline 提供多种方式来创建管道：

- 使用 AWS Command Line Interface (CLI)，其中提供方便您使用的模板。有关更多信息，请参阅使用 CLI 从 Data Pipeline 模板创建管道。
- 通过 AWS Command Line Interface (CLI) 使用 JSON 格式的管道定义文件。
- 使用 AWS 软件开发工具包和特定语言的 API。有关更多信息，请参阅使用 API。

# 使用 CLI 从 Data Pipeline 模板创建管道

Data Pipeline 提供了多个预配置的管道定义（称为模板）。您可以利用这些模板快速开始使用 AWS Data Pipeline。这些模板可在位于 Amazon S3 位置：`s3://datapipeline-us-east-1/templates/` 的公共存储桶中找到。创建这些预定义模板是为了实现特定的使用案例，可用于创建管道。您可以使用 `aws s3 ls --recursive "s3://datapipeline-us-east-1/templates/"` 来列出所有可用的模板。

## 使用 CLI 从模板创建管道

假设您想要创建一个将 DynamoDB 表导出到 Amazon S3 的管道。在这种情况下要使用的模板可以在此找到：`s3://datapipeline-us-east-1/templates/DynamoDB Templates/Export DynamoDB table to S3.json`。

使用 CLI 下载模板 JSON 并创建管道

1. 使用 `aws s3 cp` CLI 或 curl 下载模板。例如：

```
aws s3 cp "s3://datapipeline-us-east-1/templates/DynamoDB Templates/Export DynamoDB
 table to S3.json" <destination directory>
```

2. 根据需要，对已下载的模板进行更改。例如，要使用最新的 EMR 发行版本，请更改 `EmrClusterForBackup` 对象中的 `releaseLabel` 字段，更改主实例和核心实例类型，以及更改模板中参数的默认值。
3. 使用 `create-pipeline` CLI 创建管道。例如：

```
aws datapipeline create-pipeline --name my-ddb-backup-pipeline --unique-id my-ddb-
backup-pipeline --region ap-northeast-1
```

4. 记下已创建的管道 ID。
5. 使用 `put-pipeline-definition` 上传定义。使用 `--parameter-values` 选项提供要覆盖其默认值的参数的值。

有关模板的更多信息，请参阅Choose a template。

## Choose a template

以下模板可从 Amazon S3 存储桶：`s3://datapipeline-us-east-1/templates/` 下载。

模板

- ShellCommandActivity 使用入门
- 运行 AWS CLI 命令
- 将 DynamoDB 表导出到 S3
- 从 S3 导入 DynamoDB 备份数据
- 在 Amazon EMR 集群上运行作业
- 将 Amazon RDS MySQL 表完整复制到 Amazon S3
- 将 Amazon RDS MySQL 表增量复制到 Amazon S3
- 将 S3 数据加载到 Amazon RDS MySQL 表中
- 将 Amazon RDS MySQL 表完整复制到 Amazon Redshift
- 将 Amazon RDS MySQL 表增量复制到 Amazon Redshift
- 将数据从 Amazon S3 加载到 Amazon Redshift

## ShellCommandActivity 使用入门

Getting Started using ShellCommandActivity 模板运行 shell 命令脚本来计算日志文件中的 GET 请求数量。每次计划运行管道时，输出写入到带有时间戳的 Amazon S3 位置。

模板使用以下管道对象：

- ShellCommandActivity
- S3InputNode
- S3OutputNode
- Ec2Resource

## 运行 AWS CLI 命令

此模板按计划的间隔运行用户指定的 AWS CLI 命令。

## 将 DynamoDB 表导出到 S3

将 DynamoDB 表导出至 S3 模板计划利用 Amazon EMR 集群将数据从 DynamoDB 表导出到 Amazon S3 存储桶。此模板使用 Amazon EMR 集群，该集群的大小会被调整为与 DynamoDB 表的吞吐量值成

比例。虽然您可以增加表上的 IOP，但这可能会在导入和导出时产生额外的成本。以前导出操作使用 HiveActivity，但现在使用原生的 MapReduce。

模板使用以下管道对象：

- EmrActivity
- EmrCluster
- DynamoDBDataNode
- S3DataNode

## 从 S3 导入 DynamoDB 备份数据

从 S3 导入 DynamoDB 备份数据模板计划利用 Amazon EMR 集群将 Amazon S3 中以前创建的 DynamoDB 备份加载到 DynamoDB 表中。DynamoDB 表中的现有项将被备份数据项所更新，新项将添加到表中。此模板使用 Amazon EMR 集群，该集群的大小会被调整为与 DynamoDB 表的吞吐量值成比例。虽然您可以增加表上的 IOP，但这可能会在导入和导出时产生额外的成本。以前导入操作使用 HiveActivity，但现在使用原生的 MapReduce。

模板使用以下管道对象：

- EmrActivity
- EmrCluster
- DynamoDBDataNode
- S3DataNode
- S3PrefixNotEmpty

## 在 Amazon EMR 集群上运行作业

在 Elastic MapReduce 群集中运行任务模板基于提供的参数启动 Amazon EMR 集群，并基于指定的计划开始运行步骤。任务完成后，EMR 集群终止。(可选) 可以指定引导操作在集群上安装其他软件或更改应用程序配置。

模板使用以下管道对象：

- EmrActivity
- EmrCluster

## 将 Amazon RDS MySQL 表完整复制到 Amazon S3

将 RDS MySQL 表完整复制到 S3 模板完整复制 Amazon RDS MySQL 表并将输出存储在 Amazon S3 位置中。输出以 CSV 文件格式存储在指定 Amazon S3 位置下带有时间戳的子文件夹中。

模板使用以下管道对象：

- CopyActivity
- Ec2Resource
- SqlDataNode
- S3DataNode

## 将 Amazon RDS MySQL 表增量复制到 Amazon S3

将 RDS MySQL 表增量复制到 S3 模板对 Amazon RDS MySQL 表的数据执行增量复制，并将输出存储在 Amazon S3 位置中。Amazon RDS MySQL 表必须具有“上次修改日期”列。

此模板将复制在计划的间隔之间从计划开始时间开始对表所做的更改。计划类型为时间序列，因此如果在特定小时计划了复制，当表行的上次修改时间戳在该小时内时，AWS Data Pipeline 会复制这些表行。不复制对表的物理删除。每次计划运行时，输出写入到 Amazon S3 位置下带有时间戳的子文件夹中。

模板使用以下管道对象：

- CopyActivity
- Ec2Resource
- SqlDataNode
- S3DataNode

## 将 S3 数据加载到 Amazon RDS MySQL 表中

将 S3 数据加载到 RDS MySQL 表中模板计划 Amazon EC2 实例将 CSV 文件从下面指定的 Amazon S3 文件路径复制到 Amazon RDS MySQL 表。CSV 文件不应具有标头行。该模板使用 Amazon S3 数据中的条目更新 Amazon RDS MySQL 表中的现有条目，并将新条目从 Amazon S3 数据添加到 Amazon RDS MySQL 表。您可以将数据加载到现有表中，或者提供 SQL 查询来创建新表。

模板使用以下管道对象：

- CopyActivity
- Ec2Resource
- SqlDataNode
- S3DataNode

## Amazon RDS 到 Amazon Redshift 模板

以下两个模板使用转换脚本，将表从 Amazon RDS MySQL 复制到 Amazon Redshift，该脚本使用源表架构创建 Amazon Redshift 表，有以下几点需要注意：

- 如果未指定分配键，则将 Amazon RDS 表的第一个主键设置为分配键。
- 执行复制到 Amazon Redshift 的操作时，您无法跳过 Amazon RDS MySQL 表中存在的列。
- （可选）您可以提供 Amazon RDS MySQL 到 Amazon Redshift 列数据类型映射，作为模板中的参数之一。如果指定此项，则脚本将用它来创建 Amazon Redshift 表。

如果使用的是 `Overwrite_Existing` Amazon Redshift 插入模式：

- 如果未提供分配键，则将使用 Amazon RDS MySQL 表上的主键。
- 如果表上有复合主键，则在未提供分配键时，将第一个复合主键用作分配键。仅将第一个复合键设置为 Amazon Redshift 表中的主键。
- 如果未提供分配键并且 Amazon RDS MySQL 表上没有主键，则复制操作将失败。

有关 Amazon Redshift 的更多信息，请参阅以下主题：

- Amazon Redshift 集群
- Amazon Redshift COPY
- 分配方式和 DISTKEY 示例
- 排序键

下表介绍了脚本如何转换数据类型：

MySQL 与 Amazon Redshift 之间的数据类型转换

| MySQL 数据类型 | Amazon Redshift 数据类型 | 备注 |
| --- | --- | --- |
| TINYINT,<br><br>TINYINT (size) | SMALLINT | MySQL：-128 至 127。可在括号中指定的最大位数。<br><br>Amazon Redshift: INT2. 有符号的二字节整数 |
| TINYINT UNSIGNED，<br><br>TINYINT (size) UNSIGNED | SMALLINT | MySQL：0 到 255，无符号。可在括号中指定的最大位数。<br><br>Amazon Redshift: INT2. 有符号的二字节整数 |
| SMALLINT,<br><br>SMALLINT(size) | SMALLINT | MySQL：-32768 到 32767 正常。可在括号中指定的最大位数。<br><br>Amazon Redshift: INT2. 有符号的二字节整数 |
| SMALLINT UNSIGNED，<br><br>SMALLINT(size) UNSIGNED， | INTEGER | MySQL：0 到 65535，无符号*。可在括号中指定的最大位数<br><br>Amazon Redshift: INT4. 有符号的四字节整数 |
| MEDIUMINT,<br><br>MEDIUMINT(size) | INTEGER | MySQL：388608 至 8388607。可在括号中指定的最大位数<br><br>Amazon Redshift: INT4. 有符号的四字节整数 |
| MEDIUMINT UNSIGNED，<br><br>MEDIUMINT(size)<br><br>UNSIGNED | INTEGER | MySQL：0 至 16777215。可在括号中指定的最大位数 |

| MySQL 数据类型 | Amazon Redshift 数据类型 | 备注 |
| --- | --- | --- |
| | | Amazon Redshift: INT4. 有符号的四字节整数 |
| INT,<br><br>INT(size) | INTEGER | MySQL：147483648 至 2147483647<br><br>Amazon Redshift: INT4. 有符号的四字节整数 |
| INT UNSIGNED，<br><br>INT(size) UNSIGNED | BIGINT | MySQL：0 至 4294967295<br><br>Amazon Redshift: INT8. 有符号的八字节整数 |
| BIGINT<br><br>BIGINT(size) | BIGINT | Amazon Redshift: INT8. 有符号的八字节整数 |
| BIGINT UNSIGNED<br><br>BIGINT(size) UNSIGNED | VARCHAR(20*4) | MySQL：0 至 18446744073709551615<br><br>Amazon Redshift：无本地等效类型，因此使用字符数组。 |
| FLOAT<br><br>FLOAT(size,d)<br><br>FLOAT(size,d) UNSIGNED | REAL | 可在 size 参数中指定的最大位数。d 参数指定小数点右侧的最大位数。<br><br>Amazon Redshift: FLOAT4 |
| DOUBLE(size,d) | DOUBLE PRECISION | 可在 size 参数中指定的最大位数。d 参数指定小数点右侧的最大位数。<br><br>Amazon Redshift: FLOAT8 |

| MySQL 数据类型 | Amazon Redshift 数据类型 | 备注 |
| --- | --- | --- |
| DECIMAL(size,d) | DECIMAL(size,d) | DOUBLE 作为字符串存储，允许固定小数点。可在 size 参数中指定的最大位数。d 参数指定小数点右侧的最大位数。<br><br>Amazon Redshift：无本地等效类型。 |
| CHAR(size) | VARCHAR(size*4) | 保留固定长度字符串，可以包含字母、数字和特殊字符。固定大小指定为参数并用括号括起来。最多可存储 255 个字符。<br><br>右侧使用空格填补。<br><br>Amazon Redshift：CHAR 数据类型不支持多字节字符，因此使用 VARCHAR。<br><br>根据 RFC3629，每个字符的最大字节位数为 4，这将字符表限制为 U+10FFFF。 |
| VARCHAR(size) | VARCHAR(size*4) | 最多可存储 255 个字符。<br><br>VARCHAR 不支持以下无效 UTF-8 代码点：0xD800 - 0xDFFF，(字节序列：ED A0 80 - ED BF BF)，0xFDD0 - 0xFDEF，0xFFFE 和 0xFFFF，(字节序列：EF B7 90 - EF B7 AF，EF BF BE 和 EF BF BF) |
| TINYTEXT | VARCHAR(255*4) | 存储最大长度为 255 个字符的字符串 |

| MySQL 数据类型 | Amazon Redshift 数据类型 | 备注 |
| --- | --- | --- |
| TEXT | VARCHAR(max) | 存储最大长度为 65535 个字符的字符串。 |
| MEDIUMTEXT | VARCHAR(max) | 0 到 16777215 个字符 |
| LONGTEXT | VARCHAR(max) | 0 到 4294967295 个字符 |
| BOOLEAN<br>BOOL<br>TINYINT(1) | BOOLEAN | MySQL：这些类型是 TINYINT (1) 的同义词。值为零视为 false。非零值将视为 true。 |
| BINARY[(M)] | varchar(255) | M 是 0 到 255 字节，固定 |
| VARBINARY(M) | VARCHAR(max) | 0-65,535 个字节 |
| TINYBLOB | VARCHAR(255) | 0-255 个字节 |
| BLOB | VARCHAR(max) | 0-65,535 个字节 |
| MEDIUMBLOB | VARCHAR(max) | 0-16,777,215 个字节 |
| LONGBLOB | VARCHAR(max) | 0-4,294,967,295 个字节 |
| ENUM | VARCHAR(255*2) | 这不是对文本枚举字符串长度的限制，而是在表定义上对枚举值数量的限制。 |
| SET | VARCHAR(255*2) | 类似于枚举。 |
| DATE | DATE | (YYYY-MM-DD)<br>“1000-01-01”到“9999-12-31” |
| TIME | VARCHAR(10*4) | (hh:mm:ss)<br>“-838-59-59”到“838-59-59” |

| MySQL 数据类型 | Amazon Redshift 数据类型 | 备注 |
| --- | --- | --- |
| DATETIME | TIMESTAMP | (YYYY-MM-DD hh:mm:ss)<br>“1000-01-01 00:00:00” 到“9999-12-31 23:59:59” |
| TIMESTAMP | TIMESTAMP | (YYYYMMDDhhmmss)<br>19700101000000 到 2037+ |
| YEAR | VARCHAR(4*4) | (YYYY)<br>1900 到 2155 |
| column SERIAL | ID 生成/OLAP 数据仓库不需要此属性，因为会复制此列。<br>SERIAL 关键字在转换时不添加。 | SERIAL 实际上是名为 SEQUENCE 的实体。它独立于您的表的其余部分存在。<br>column GENERATED BY DEFAULT<br>等效于：<br>CREATE SEQUENCE name; CREATE TABLE table ( column INTEGER NOT NULL DEFAULT nextval(name) ); |

| MySQL 数据类型 | Amazon Redshift 数据类型 | 备注 |
| --- | --- | --- |
| column BIGINT UNSIGNED NOT NULL AUTO_INCR EMENT UNIQUE | ID 生成/OLAP 数据仓库不需要此属性，因为会复制此列。<br><br>因此，SERIAL 关键字在转换时不添加。 | SERIAL 实际上是名为 SEQUENCE 的实体。它独立于您的表的其余部分存在。<br><br>column GENERATED BY DEFAULT<br><br>等效于：<br><br>CREATE SEQUENCE name; CREATE TABLE table ( column INTEGER NOT NULL DEFAULT nextval(n ame) ); |
| ZEROFILL | ZEROFILL 关键字在转换时不添加。 | INT UNSIGNED ZEROFILL NOT NULL<br><br>ZEROFILL 用零填补字段的显示值，直至达到在列定义中指定的显示宽度。超过此显示宽度的值不截断。请注意，使用 ZEROFILL 还表示 UNSIGNED。 |

## 将 Amazon RDS MySQL 表完整复制到 Amazon Redshift

将 Amazon RDS MySQL 表完整复制到 Amazon Redshift 模板通过将数据暂存在 Amazon S3 文件夹中，将完整的 Amazon RDS MySQL 表复制到 Amazon Redshift 表。Amazon S3 暂存文件夹必须与 Amazon Redshift 集群位于同一区域。如果没有 Amazon Redshift 表，则将创建与源 Amazon RDS MySQL 表具有相同架构的表。请提供在 Amazon Redshift 表创建期间，您希望应用的任意 Amazon RDS MySQL 到 Amazon Redshift 列数据类型覆盖。

模板使用以下管道对象：

- CopyActivity

- RedshiftCopyActivity
- S3DataNode
- SqlDataNode
- RedshiftDataNode
- RedshiftDatabase

## 将 Amazon RDS MySQL 表增量复制到 Amazon Redshift

增量复制 Amazon RDS MySQL 表到 Amazon Redshift 模板通过将数据暂存在 Amazon S3 文件夹中，将 Amazon RDS MySQL 表的数据复制到 Amazon Redshift 表。

Amazon S3 暂存文件夹必须与 Amazon Redshift 集群位于同一区域。

如果没有 Amazon Redshift 表，AWS Data Pipeline 将使用转换脚本创建与源 Amazon RDS MySQL 表具有相同架构的表。您必须提供在 Amazon Redshift 表创建期间，您希望应用的任意 Amazon RDS MySQL 到 Amazon Redshift 列数据类型覆盖。

此模板将复制在计划间隔之间从计划开始时间开始对 Amazon RDS MySQL 表所做的更改。不复制对 Amazon RDS MySQL 表的物理删除。您必须提供存储上次修改时间值的列名。

当您使用默认模板来为增量 Amazon RDS 复制创建管道时，将创建一个具有默认名称 `RDSToS3CopyActivity` 的活动。您可以重命名该活动。

模板使用以下管道对象：

- CopyActivity
- RedshiftCopyActivity
- S3DataNode
- SqlDataNode
- RedshiftDataNode
- RedshiftDatabase

## 将数据从 Amazon S3 加载到 Amazon Redshift

将 S3 中的数据加载到 Redshift 模板将数据从 Amazon S3 文件夹复制到 Amazon Redshift 表中。您可以将数据加载到现有表中，或者提供 SQL 查询以创建表。

根据 Amazon Redshift COPY 选项复制数据。Amazon Redshift 表必须与 Amazon S3 中的数据具有相同架构。有关 COPY 选项，请参阅 Amazon Redshift 数据库开发人员指南中的 COPY。

模板使用以下管道对象：

- CopyActivity
- RedshiftCopyActivity
- S3DataNode
- RedshiftDataNode
- RedshiftDatabase
- Ec2Resource

## 使用参数化模板创建管道

您可以使用参数化模板自定义管道定义。这使您可以创建通用管道定义，但在将管道定义添加到新管道时可提供不同参数。

目录



### 添加 myVariables 到管道定义

创建管道定义文件时，使用以下语法指定变量：#{my*Variable*}。变量需要使用前缀 my。例如，管道定义文件 pipeline-definition.json 包括以下变量：*myShellCmd*、*myS3InputLoc* 和 *myS3OutputLoc*。

> **Note**
>
> 管道定义的上限为 50 个参数。

```
{
  "objects": [
    {
```

```
      "id": "ShellCommandActivityObj",
      "input": {
        "ref": "S3InputLocation"
      },
      "name": "ShellCommandActivityObj",
      "runsOn": {
        "ref": "EC2ResourceObj"
      },
      "command": "#{myShellCmd}",
      "output": {
        "ref": "S3OutputLocation"
      },
      "type": "ShellCommandActivity",
      "stage": "true"
    },
    {
      "id": "Default",
      "scheduleType": "CRON",
      "failureAndRerunMode": "CASCADE",
      "schedule": {
        "ref": "Schedule_15mins"
      },
      "name": "Default",
      "role": "DataPipelineDefaultRole",
      "resourceRole": "DataPipelineDefaultResourceRole"
    },
    {
      "id": "S3InputLocation",
      "name": "S3InputLocation",
      "directoryPath": "#{myS3InputLoc}",
      "type": "S3DataNode"
    },
    {
      "id": "S3OutputLocation",
      "name": "S3OutputLocation",
      "directoryPath": "#{myS3OutputLoc}/#{format(@scheduledStartTime, 'YYYY-MM-dd-HH-
mm-ss')}",
      "type": "S3DataNode"
    },
    {
      "id": "Schedule_15mins",
      "occurrences": "4",
      "name": "Every 15 minutes",
      "startAt": "FIRST_ACTIVATION_DATE_TIME",
```

```
      "type": "Schedule",
      "period": "15 Minutes"
    },
    {
      "terminateAfter": "20 Minutes",
      "id": "EC2ResourceObj",
      "name": "EC2ResourceObj",
   "instanceType":"t1.micro",
      "type": "Ec2Resource"
    }
  ]
}
```

定义参数对象

您可以使用定义您管道定义中变量的参数对象，创建一个单独的文件。例如，JSON 文件 parameters.json 包含来自以上示例管道定义的 *myShellCmd*、*myS3InputLoc* 和 *myS3OutputLoc* 变量的参数对象。

```
{
  "parameters": [
    {
      "id": "myShellCmd",
      "description": "Shell command to run",
      "type": "String",
      "default": "grep -rc \"GET\" ${INPUT1_STAGING_DIR}/* > ${OUTPUT1_STAGING_DIR}/
output.txt"
    },
    {
      "id": "myS3InputLoc",
      "description": "S3 input location",
      "type": "AWS::S3::ObjectKey",
      "default": "s3://us-east-1.elasticmapreduce.samples/pig-apache-logs/data"
    },
    {
      "id": "myS3OutputLoc",
      "description": "S3 output location",
      "type": "AWS::S3::ObjectKey"
    }
  ]
}
```

> **Note**
>
> 您可以直接向管道定义文件添加这些对象，而不是使用单独的文件。

下表介绍参数对象的属性。

参数属性

| 属性 | 类型 | 描述 |
| --- | --- | --- |
| id | 字符串 | 参数的唯一标识符。要在键入或显示值时遮掩这些值，请添加星号 (“*”) 作为前缀。例如，*myVariable 。请注意，这还会在存储之前加密 AWS Data Pipeline 中的值。 |
| description | 字符串 | 参数的说明。 |
| type | String、Integer、Double 或 AWS::S3::ObjectKey | 定义输入值的允许范围以及验证规则的参数类型。默认值为 String。 |
| 可选 | 布尔值 | 指示参数可选还是必需。默认为 false。 |
| allowedValues | 字符串列表 | 枚举参数所有允许的值。 |
| 默认值 | 字符串 | 参数的默认值。如果您使用参数值指定此参数的值，它会覆盖默认值。 |
| isArray | 布尔值 | 指示参数是否为数组。 |

定义参数值

您可以创建单独的文件，使用参数值来定义变量。例如，JSON 文件 file://values.json 包含来自以上示例管道定义的 *myS3OutputLoc* 变量的值。

```
{
  "values":
    {
      "myS3OutputLoc": "myOutputLocation"
    }
}
```

提交管道定义

当您提交管道定义时，您可以指定参数、参数对象和参数值。例如，您可以使用 put-pipeline-definition AWS CLI 命令，如下所示：

```
$ aws datapipeline put-pipeline-definition --pipeline-id id --pipeline-definition
 file://pipeline-definition.json \
--parameter-objects file://parameters.json --parameter-values-uri file://values.json
```

> **Note**
>
> 管道定义的上限为 50 个参数。parameter-values-uri 文件大小的上限为 15 KB。

# 查看管道

您可以使用命令行界面 (CLI) 查看您的管道。

使用 AWS CLI 查看管道

- 使用以下 list-pipelines 命令列出您的管道：

  ```
  aws datapipeline list-pipelines
  ```

## 解释管道状态代码

AWS Data Pipeline 控制台和 CLI 中显示的状态级别指示管道及其组件的状况。管道状态仅仅是管道的概览；要查看更多信息，请查看单个管道组件的状态。

管道在就绪 (管道定义通过验证)、当前正在执行工作或者完成执行工作时具有 SCHEDULED 状态。管道在未激活或者无法执行工作 (例如，管道定义未通过验证) 时具有 PENDING 状态。

管道的状态为 PENDING、INACTIVE 或 FINISHED 时，将其视为不活动。不活动管道会产生费用 (有关更多信息，请参阅定价)。

状态代码

ACTIVATING

组件或资源正在启动，例如 EC2 实例。

CANCELED

该组件在可以运行之前已被用户或 AWS Data Pipeline 取消。当该组件所依赖的其他组件或资源发生故障时，可能会自动发生这种情况。

CASCADE_FAILED

该组件或资源因其依赖项之一的级联故障而被取消，但该组件可能不是故障的原始来源。

DEACTIVATING

管道正在停用。

FAILED

组件或资源遇到错误并停止了工作。当某个组件或资源出现故障时，可能会导致取消和故障级联到依赖该组件的其他组件。

FINISHED

该组件完成了其分配到的工作。

INACTIVE

管道已停用。

PAUSED

该组件已暂停，目前未执行其工作。

PENDING

管道已准备好第一次激活。

RUNNING

资源正在运行并准备好接收工作。

SCHEDULED

资源已计划运行。

SHUTTING_DOWN

资源在成功完成其工作后正在关闭。

SKIPPED

在激活管道后，该组件使用晚于当前计划的时间戳跳过了执行间隔。

TIMEDOUT

该资源超过了terminateAfter阈值并已被AWS Data Pipeline停止。资源达到此状态后，AWS Data Pipeline忽略该资源的actionOnResourceFailure、retryDelay和retryTimeout值。该状态仅适用于资源。

VALIDATING

AWS Data Pipeline 正在验证管道定义。

WAITING_FOR_RUNNER

该组件正在等待其工作客户端检索工作项。组件和工作客户端关系由该组件定义的 runsOn 或 workerGroup 字段控制。

WAITING_ON_DEPENDENCIES

在执行其工作之前，该组件正在验证其默认和用户配置的前提条件是否得到满足。

## 解释管道及其组件运行状况

管道中的每个管道和组件返回运行状况 HEALTHY、ERROR、"-"、No Completed Executions 或 No Health Information Available。只有在管道组件完成了首次执行或者组件先决条件失败时，管道才具有运行状况。组件的运行状况聚合为管道运行状况，在您查看管道执行详细信息时，首先将看到错误状态。

管道运行状况

HEALTHY

所有组件的聚合运行状况为 HEALTHY。这意味着必须至少有一个组件已成功完成。在HEALTHY执行详细信息页面上，您可以单击 状态查看最近成功完成的管道组件实例。

ERROR

管道中至少有一个组件的运行状况为 ERROR。在 ERRORExecution Details 页面上，您可以单击 状态查看最近失败的管道组件实例。

No Completed Executions 或 No Health Information Available。

未报告此管道的任何运行状况。

> **Note**
>
> 虽然组件几乎立即更新其运行状况，但最多可能需要五分钟时间来更新管道运行状况。

组件运行状况

HEALTHY

组件 (Activity 或 DataNode) 如果已经成功完成了执行、其状态标记为 FINISHED 或 MARK_FINISHED，则运行状况为 HEALTHY。在HEALTHY执行详细信息页面上，您可以单击组件的名称或 状态查看最近成功完成的管道组件实例。

ERROR

在组件级别出错或者其先决条件之一失败。状态为 FAILED、TIMEOUT 或 CANCELED 会触发此错误。在 ERROR Execution Details 页面上，您可以单击组件的名称或 状态查看最近失败的管道组件实例。

No Completed Executions 或 No Health Information Available

未报告此组件的任何运行状况。

## 查看管道定义

使用命令行界面 (CLI) 查看管道定义。CLI 以 JSON 格式输出管道定义文件。有关管道定义文件的语法和用法的信息，请参阅管道定义文件语法。

当使用 CLI，一种好的做法是先检索管道定义，然后提交修改，因为在您上次处理之后，可能会有其他用户或进程更改了管道定义。通过下载当前定义的副本并以此为基础来进行修改，您可以确保使用最新的管道定义。在修改之后，重新检索管道定义也是一种好的做法，这样您可以确保更新成功。

当使用的是 CLI，则可以获取管道的两个不同版本。active 版本是当前正在运行的管道。latest 版本是您编辑正在运行的管道时创建的副本。当您上传编辑后的管道时，它会成为 active 版本，以前的 active 版本不再可用。

使用 AWS CLI 获取管道定义

要获取完整的管道定义，请使用 get-pipeline-definition 命令。管道定义输出到标准输出 (stdout)。

以下示例获取指定管道的管道定义。

```
aws datapipeline get-pipeline-definition --pipeline-id df-00627471SOVYZEXAMPLE
```

要检索管道的特定版本，请使用 `--version` 选项。下面的示例检索指定管道的 `active` 版本。

```
aws datapipeline get-pipeline-definition --version active --id df-00627471SOVYZEXAMPLE
```

## 查看管道实例详细信息

您可以监控管道的进度。有关实例状态的更多信息，请参阅 解释管道状态详细信息。有关管道的实例运行失败或未完成的问题排查的更多信息，请参阅解决常见问题。

使用 AWS CLI 监控管道的进度

要检索管道实例详细信息，例如管道已经运行次数的历史记录，请使用 list-runs 命令。此命令使您可以基于运行的当前状态或启动的日期范围，筛选返回的运行列表。筛选结果非常有用，因为根据管道的期限和计划，其运行历史记录可能会非常大。

以下示例检索所有运行的信息。

```
aws datapipeline list-runs --pipeline-id df-00627471SOVYZEXAMPLE
```

以下示例检索所有已完成运行的信息。

```
aws datapipeline list-runs --pipeline-id df-00627471SOVYZEXAMPLE --status finished
```

以下示例检索在指定时间范围内启动的所有运行的信息。

```
aws datapipeline list-runs --pipeline-id df-00627471SOVYZEXAMPLE --start-interval
 "2013-09-02","2013-09-11"
```

## 查看管道日志

管道创建中支持管道级别的日志记录，其方法是在控制台中指定 Amazon S3 位置，或者使用开发工具包/CLI 的默认对象中的 `pipelineLogUri`。该 URI 中每个管道的目录结构类似于下文：

```
pipelineId
    -componentName
        -instanceId
            -attemptId
```

对于管道 df-00123456ABC7DEF8HIJK，目录结构如下所示：

```
df-00123456ABC7DEF8HIJK
    -ActivityId_fXNzc
        -@ActivityId_fXNzc_2014-05-01T00:00:00
            -@ActivityId_fXNzc_2014-05-01T00:00:00_Attempt=1
```

对于 ShellCommandActivity，与这些活动关联的 stderr 和 stdout 的日志存储在每次尝试的目录中。

对于类似于 EmrCluster 的资源，其中设置了 emrLogUri，该值优先。否则，资源 (包括这些资源的 TaskRunner 日志) 将遵循上述管道日志记录结构。

要查看给定管道运行的日志，请执行以下操作：

1. 通过调用 query-objects 获取确切的对象 ID 来检索 ObjectId。例如：

   ```
   aws datapipeline query-objects --pipeline-id <pipeline-id> --sphere ATTEMPT --region
    ap-northeast-1
   ```

   query-objects 是一个分页的 CLI，如果给定的 pipeline-id 执行次数更多，则可能会返回分页标记。您可以使用该标记完成所有尝试，直到找到预期的对象。例如，返回的 ObjectId 将如下所示：@TableBackupActivity_2023-05-020T18:05:18_Attempt=1。
2. 使用 ObjectId，使用以下方法检索日志位置：

   ```
   aws datapipeline describe-objects –pipeline-id <pipeline-id> --object-ids <object-id>
    --query "pipelineObjects[].fields[?key=='@logLocation'].stringValue"
   ```

失败活动的错误消息

要获取错误消息，请先使用 query-objects 获取 ObjectId。

检索失败的 ObjectId 后，使用 describe-objects CLI 获取实际的错误消息。

```
aws datapipeline describe-objects --region ap-northeast-1 --pipeline-id
 <pipeline-id> --object-ids <object-id> --query "pipelineObjects[].fields[?
key=='errorMessage'].stringValue"
```

取消或重新运行对象或将对象标记为已完成

使用 `set-status` CLI 取消正在运行的对象，或者重新运行失败的对象或将正在运行的对象标记为已完成。

首先，使用 `query-objects` CLI 获取对象 ID。例如：

```
aws datapipeline query-objects --pipeline-id <pipeline-id> --sphere INSTANCE --region
 ap-northeast-1
```

使用 `set-status` CLI 更改所需对象的状态。例如：

```
aws datapipeline set-status –pipeline-id <pipeline-id> --region ap-northeast-1 --status
 TRY_CANCEL --object-ids <object-id>
```

# 编辑管道

如果您需要更改管道之一的某个方面，可以更新其管道定义。在更改了正在运行的管道之后，您必须重新激活该管道以使更改生效。此外，您可以重新运行一个或多个管道组件。

目录



## 限制

在管道处于 PENDING 状态且未激活时，您不可以对它进行任何更改。激活管道之后，您可以编辑管道，但有以下限制。在您保存所做的更改，然后重新激活管道之后，更改会应用到管道对象的新运行。

- 您无法删除对象
- 您无法更改现有对象的计划周期
- 您无法添加、删除或修改现有对象中的引用字段
- 您无法在新对象的输出字段中引用现有对象

- 您无法更改对象的计划开始日期 (改为使用特定日期和时间激活管道)

## 使用 AWS CLI 编辑管道

您可以使用命令行工具编辑管道。

首先，请使用 get-pipeline-definition 命令下载当前管道定义的副本。通过执行此操作，您可以确保修改的是最新的管道定义。以下示例将管道定义输出到标准输出 (stdout)。

```
aws datapipeline get-pipeline-definition --pipeline-id df-00627471SOVYZEXAMPLE
```

将管道定义保存到文件并根据需要进行编辑。使用 put-pipeline-definition 命令更新管道定义。以下示例上传更新后的管道定义文件。

```
aws datapipeline put-pipeline-definition --pipeline-id df-00627471SOVYZEXAMPLE --
pipeline-definition file://MyEmrPipelineDefinition.json
```

您可以使用 `get-pipeline-definition` 命令重新检索管道定义，确保更新成功。要激活管道，请使用以下 activate-pipeline 命令：

```
aws datapipeline activate-pipeline --pipeline-id df-00627471SOVYZEXAMPLE
```

如果愿意，您可以使用 `--start-timestamp` 选项，从特定的日期和时间激活管道，如下所示：

```
aws datapipeline activate-pipeline --pipeline-id df-00627471SOVYZEXAMPLE --start-
timestamp YYYY-MM-DDTHH:MM:SSZ
```

要重新运行一个或多个管道组件，请使用 set-status 命令。

# 克隆管道

克隆操作将生成管道的副本并允许您为新管道指定名称。您可以克隆任意状态下的管道，即使管道有错误；但是，新管道会保持 `PENDING` 状态，直至您手动激活它。对于新管道，克隆操作使用原始管道定义的最新版本而不是活动的版本。在克隆操作中，原始管道的完整计划不复制到新管道，仅复制周期设置。

使用 AWS CLI 克隆管道：

1. 使用新名称和唯一 ID 来创建新管道。记下返回的管道 ID。

2. 使用 get-pipeline-definition CLI 获取要克隆的现有管道的管道定义，并将其写入临时文件。记下该文件的绝对路径。
3. 使用 put-pipeline-definition CLI 将管道定义从现有管道复制到新管道。
4. 使用 get-pipeline-definition CLI 获取新管道的定义以验证管道定义。

```
# Create Pipeline (returns <new-pipeline-id>)
aws datapipeline create-pipeline --name my-cloned-pipeline --unique-id my-cloned-
pipeline --region ap-northeast-1

#Get pipeline definition of existing pipeline
aws datapipeline get-pipeline-definition --pipeline-id <existing-pipeline-id> --
region ap-northeast-1 > existing_pipeline_definition.json

# Put pipeline definition to new pipeline
aws datapipeline put-pipeline-definition --pipeline-id <new-
pipeline-id> --region ap-northeast-1 --pipeline-definition file://
<absolute_path_to_existing_pipeline_definition.json>

# get pipeline definition of new pipeline
aws datapipeline get-pipeline-definition --pipeline-id <new-pipeline-id> --region
 ap-northeast-1
```

# 标记管道

标签是区分大小写的键/值对，由一个键和一个可选值组成，均由用户定义。您最多可以将 10 个标签应用于每个管道。每个管道的标签键必须唯一。如果您添加的标签中的键已经与管道关联，它将更新该标签的值。

将标签应用到管道还会将标签传播到其基础资源（例如，Amazon EMR 群集和 Amazon EC2 实例）。但是，它不会将这些标签应用到处于 FINISHED 状态或已终止状态的资源。如果需要，您可以使用 CLI 将标签应用于这些资源。

当您完成使用标签后，可以从管道中将其删除。

使用 AWS CLI 标记管道

要将标签添加到新管道，请将 --tags 选项添加到 create-pipeline 命令。例如，以下选项创建一个管道，带有两个标签，environment 标签的值为 production，owner 标签的值为 sales。

```
--tags key=environment,value=production key=owner,value=sales
```

要将标签添加到现有管道，请使用 add-tags 命令，如下所示：

```
aws datapipeline add-tags --pipeline-id df-00627471SOVYZEXAMPLE --tags
 key=environment,value=production key=owner,value=sales
```

要从现有管道中删除标签，请使用 remove-tags 命令，如下所示：

```
aws datapipeline remove-tags --pipeline-id df-00627471SOVYZEXAMPLE --tag-keys
 environment owner
```

# 停用管道

停用正在运行的管道将暂停管道执行。要恢复管道执行，您可以激活管道。这使您能够进行更改。例如，如果您正在将数据写入计划要进行维护的数据库，您可以停用管道，等待维护完成，然后激活管道。

在您停用管道时，您可以指定对正在运行活动执行的操作。默认情况下，这些活动将立即取消。或者，您可以让 AWS Data Pipeline 等待直至活动完成，然后再停用管道。

在您激活已停用的管道时，您可以指定其恢复时间。使用 AWS CLI 或 API，默认情况下管道从上次完成的执行恢复，或者您可以指定恢复管道的日期和时间。

## 使用 AWS CLI 停用管道

使用以下 deactivate-pipeline 命令停用管道：

```
aws datapipeline deactivate-pipeline --pipeline-id df-00627471SOVYZEXAMPLE
```

要仅在所有正在运行的活动完成后停用管道，请添加 `--no-cancel-active` 选项，如下所示：

```
aws datapipeline deactivate-pipeline --pipeline-id df-00627471SOVYZEXAMPLE --no-cancel-
active
```

在您准备好之后，您可以使用以下 activate-pipeline 命令，从上次停用的位置恢复管道执行：

```
aws datapipeline activate-pipeline --pipeline-id df-00627471SOVYZEXAMPLE
```

要从特定日期和时间启动管道，请添加 `--start-timestamp` 选项，如下所示：

```
aws datapipeline activate-pipeline --pipeline-id df-00627471SOVYZEXAMPLE --start-timestamp YYYY-MM-DDTHH:MM:SSZ
```

## 删除管道

当您不再需要某个管道时，例如应用程序测试期间创建的管道，您应删除该管道以将其从活动使用中移除。删除管道会将其置于“deleting”状态。当管道处于“deleted”状态时，其管道定义和运行历史记录均已被删除。因此，您无法对该管道继续执行操作，包括描述该管道。

> **Important**
>
> 在管道删除之后，您无法恢复它，因此在删除之前，请确保您以后不再需要该管道。

使用 AWS CLI 删除管道

要删除管道，请使用 delete-pipeline 命令。以下命令删除指定管道。

```
aws datapipeline delete-pipeline --pipeline-id df-00627471SOVYZEXAMPLE
```

## 将数据和表与管道活动一起暂存

AWS Data Pipeline 可以暂存您的管道中的输入和输出数据，以便更容易使用特定活动，例如 `ShellCommandActivity` 和 `HiveActivity`。

数据暂存让您能够将数据从输入数据节点复制到执行活动的资源，从资源到输出数据节点与此类似。

通过在活动的 shell 命令或 Hive 脚本中使用特殊变量，可使用 Amazon EMR 或 Amazon EC2 资源上的暂存数据。

表暂存类似于数据暂存，具体而言，不同之处在于暂存的数据采用数据库表的形式。

AWS Data Pipeline 支持以下暂存场景：

- 使用 `ShellCommandActivity` 的数据暂存

- 使用 Hive 的表暂存和支持暂存的数据节点
- 使用 Hive 的表暂存和不支持暂存的数据节点

> **Note**
>
> 仅当活动上的 `stage` 字段设置为 `true` 时暂存才生效，例如 `ShellCommandActivity`。有关更多信息，请参阅ShellCommandActivity。

此外，数据节点和活动可以通过四种方式关联：

在资源上本地暂存数据

输入数据自动复制到资源本地文件系统。输出数据自动从资源本地文件系统复制到输出数据节点。例如，当您使用 staging = true 配置 `ShellCommandActivity` 输入和输出时，输入数据作为 INPUTx_STAGING_DIR 可用，输出数据作为 OUTPUTx_STAGING_DIR 可用，其中 x 是输入或输出的编号。

暂存活动的输入和输出定义

输入数据格式 (列名和表名) 自动复制到活动的资源中。例如，当您使用 staging = true 配置 `HiveActivity` 时。在输入 `S3DataNode` 上指定的数据格式用于从 Hive 表暂存表定义。

暂存未启用

输入和输出对象及其字段可用于活动，但数据本身不行。例如，`EmrActivity` 默认情况下或在您使用 staging = false 配置其他活动时。在此配置中，数据字段可供活动使用 AWS Data Pipeline 表达式语法引用它们，这仅在满足依赖关系时发生。这仅用作依赖关系检查。活动中的代码负责将数据从输入复制到运行活动的资源。

对象之间的依赖关系

两个对象之间存在依赖关系，这会导致类似于未启用暂存的情况。这导致数据节点或活动用作执行另一个活动的先决条件。

## 使用 ShellCommandActivity 的数据暂存

请考虑使用 `ShellCommandActivity` 和 `S3DataNode` 对象作为数据输入和输出的场景。AWS Data Pipeline 自动暂存数据节点，使其可通过 shell 命令使用环境变量 `${INPUT1_STAGING_DIR}` 和 `${OUTPUT1_STAGING_DIR}` 访问，就像在本地文件夹中一样，如下例中所示。名为

INPUT1_STAGING_DIR 和 OUTPUT1_STAGING_DIR 的变量的数字部分根据您的活动引用的数据节点数递增。

> **Note**
>
> 只有在您的输入和输出为 S3DataNode 对象时，此场景才按所述工作。此外，只有当 directoryPath 设置在输出 S3DataNode 对象上时，才允许输出数据暂存。

```
{
  "id": "AggregateFiles",
  "type": "ShellCommandActivity",
  "stage": "true",
  "command": "cat ${INPUT1_STAGING_DIR}/part* > ${OUTPUT1_STAGING_DIR}/aggregated.csv",
  "input": {
    "ref": "MyInputData"
  },
  "output": {
    "ref": "MyOutputData"
  }
},
{
  "id": "MyInputData",
  "type": "S3DataNode",
  "schedule": {
    "ref": "MySchedule"
  },
  "filePath": "s3://my_bucket/source/#{format(@scheduledStartTime,'YYYY-MM-
dd_HHmmss')}/items"
  }
},
{
  "id": "MyOutputData",
  "type": "S3DataNode",
  "schedule": {
    "ref": "MySchedule"
  },
  "directoryPath": "s3://my_bucket/destination/#{format(@scheduledStartTime,'YYYY-MM-
dd_HHmmss')}"
  }
},
...
```

# 使用 Hive 的表暂存和支持暂存的数据节点

请考虑使用 HiveActivity 和 S3DataNode 对象作为数据输入和输出的场景。AWS Data Pipeline 自动暂存数据节点，使其可通过 Hive 脚本使用变量 ${input1} 和 ${output1} 访问，就像它们是 Hive 表一样，如下例 HiveActivity 中所示。名为 input 和 output 的变量的数字部分根据您的活动引用的数据节点数递增。

> **Note**
>
> 只有在您的输入和输出为 S3DataNode 或 MySqlDataNode 对象时，此场景才按所述方式工作。DynamoDBDataNode 不支持表暂存。

```
{
  "id": "MyHiveActivity",
  "type": "HiveActivity",
  "schedule": {
    "ref": "MySchedule"
  },
  "runsOn": {
    "ref": "MyEmrResource"
  },
  "input": {
    "ref": "MyInputData"
  },
  "output": {
    "ref": "MyOutputData"
  },
  "hiveScript": "INSERT OVERWRITE TABLE ${output1} select * from ${input1};"
},
{
  "id": "MyInputData",
  "type": "S3DataNode",
  "schedule": {
    "ref": "MySchedule"
  },
  "directoryPath": "s3://test-hive/input"
  }
},
{
  "id": "MyOutputData",
  "type": "S3DataNode",
```

```
  "schedule": {
    "ref": "MySchedule"
  },
  "directoryPath": "s3://test-hive/output"
  }
},
...
```

## 使用 Hive 的表暂存和不支持暂存的数据节点

请考虑使用 HiveActivity 和 DynamoDBDataNode 作为数据输入并将 S3DataNode 对象作为输出的场景。没有数据暂存可用于 DynamoDBDataNode，因此您必须先手动在 hive 脚本中创建表，使用变量名 #{input.tableName} 引用 DynamoDB 表。如果 DynamoDB 表是输出，类似术语也适用，除非您使用变量 #{output.tableName}。在本示例中，暂存可用于输出 S3DataNode 对象，因此您可以将输出数据节点作为 ${output1} 引用。

> **Note**
>
> 在本示例中，表名变量具有 # ( 井号 ) 字符前缀，因为 AWS Data Pipeline 使用表达式访问 tableName 或 directoryPath。有关表达式求值在 AWS Data Pipeline 中工作方式的详细信息，请参阅 表达式计算。

```
{
  "id": "MyHiveActivity",
  "type": "HiveActivity",
  "schedule": {
    "ref": "MySchedule"
  },
  "runsOn": {
    "ref": "MyEmrResource"
  },
  "input": {
    "ref": "MyDynamoData"
  },
  "output": {
    "ref": "MyS3Data"
  },
  "hiveScript": "-- Map DynamoDB Table
SET dynamodb.endpoint=dynamodb.us-east-1.amazonaws.com;
SET dynamodb.throughput.read.percent = 0.5;
```

```
CREATE EXTERNAL TABLE dynamodb_table (item map<string,string>)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES ("dynamodb.table.name" = "#{input.tableName}");
INSERT OVERWRITE TABLE ${output1} SELECT * FROM dynamodb_table;"
},
{
  "id": "MyDynamoData",
  "type": "DynamoDBDataNode",
  "schedule": {
    "ref": "MySchedule"
  },
  "tableName": "MyDDBTable"
},
{
  "id": "MyS3Data",
  "type": "S3DataNode",
  "schedule": {
    "ref": "MySchedule"
  },
  "directoryPath": "s3://test-hive/output"
  }
},
...
```

# 利用多个区域中的资源使用管道

默认情况下，Ec2Resource 和 EmrCluster 资源在与 AWS Data Pipeline 相同的区域中运行，但是，AWS Data Pipeline 支持在多个区域中协调数据流的功能，例如在一个区域中运行的资源会整合来自另一个区域的输入数据。通过允许资源在指定区域中运行，您还具备了灵活性，可以将资源与其从属数据集放置在一起，通过减少延迟来最大化性能，同时避免跨区域的数据传输费用。您可以在 Ec2Resource 和 EmrCluster 上使用 region 字段，配置资源在不同于 AWS Data Pipeline 的区域中运行。

以下示例管道 JSON 文件显示了如何在欧洲地区（爱尔兰）运行 EmrCluster 资源，假定集群要处理的大量数据位于相同区域上。在本示例中，与典型管道的唯一的区别是 EmrCluster 将 region 字段值设置为 eu-west-1。

```
{
  "objects": [
    {
      "id": "Hourly",
```

```
      "type": "Schedule",
      "startDateTime": "2014-11-19T07:48:00",
      "endDateTime": "2014-11-21T07:48:00",
      "period": "1 hours"
    },
    {
      "id": "MyCluster",
      "type": "EmrCluster",
      "masterInstanceType": "m3.medium",
      "region": "eu-west-1",
      "schedule": {
        "ref": "Hourly"
      }
    },
    {
      "id": "MyEmrActivity",
      "type": "EmrActivity",
      "schedule": {
        "ref": "Hourly"
      },
      "runsOn": {
        "ref": "MyCluster"
      },
      "step": "/home/hadoop/contrib/streaming/hadoop-streaming.jar,-input,s3n://
elasticmapreduce/samples/wordcount/input,-output,s3://eu-west-1-bucket/wordcount/
output/#{@scheduledStartTime},-mapper,s3n://elasticmapreduce/samples/wordcount/
wordSplitter.py,-reducer,aggregate"
    }
  ]
}
```

下表列出了您可以在 `region` 字段中选择的区域以及使用的关联区域代码：

> **Note**
>
> 以下列表包含一些区域，AWS Data Pipeline 可以在其中协调工作流以及启动 Amazon EMR 或 Amazon EC2 资源。可能在这些区域中不支持 AWS Data Pipeline。有关支持 AWS Data Pipeline 的区域的信息，请参阅 Amazon Web Services Region 和终端节点。

| 区域名称 | 区域代码 |
| --- | --- |
| 美国东部（弗吉尼亚州北部） | us-east-1 |
| US East (Ohio) | us-east-2 |
| 美国西部（北加利福尼亚） | us-west-1 |
| US West (Oregon) | us-west-2 |
| Canada (Central) | ca-central-1 |
| Europe (Ireland) | eu-west-1 |
| 欧洲（伦敦） | eu-west-2 |
| 欧洲（法兰克福） | eu-central-1 |
| 亚太地区（新加坡） | ap-southeast-1 |
| Asia Pacific (Sydney) | ap-southeast-2 |
| 亚太地区（孟买） | ap-south-1 |
| 亚太地区（东京) | ap-northeast-1 |
| 亚太地区（首尔） | ap-northeast-2 |
| South America (São Paulo) | sa-east-1 |

# 级联故障和重新运行

AWS Data Pipeline 允许您配置在依赖项失败或者由用户取消时，管道对象的行为方式。您可以确保故障级联到其他管道对象 (使用者) 以防止无限期等待。所有活动、数据节点和先决条件具有名为 `failureAndRerunMode` 的字段，其默认值为 `none`。要启用级联故障，请将 `failureAndRerunMode` 字段设置为 `cascade`。

启用此字段时，如果管道对象阻塞在 `WAITING_ON_DEPENDENCIES` 状态下，并且任何依赖项已失败且没有待处理命令，则出现级联故障。在级联故障期间，将发生以下事件：

- 对象失败时，其使用者设置为 CASCADE_FAILED，并且原始对象及其使用者的先决条件均设置为 CANCELED。
- 忽略任何已处于 FINISHED、FAILED 或 CANCELED 状态的对象。

级联故障不对失败对象的依赖项 (上游) 进行操作，除了与原始失败对象关联的先决条件。级联故障影响的管道对象可能会触发任意重试或操作后活动，例如 onFail。

级联故障的具体效果取决于对象类型。

## 活动

某个活动的任意依赖项失败时将更改为 CASCADE_FAILED，并且接下来会在该活动的使用者中触发级联故障。如果活动依赖的资源失败，该活动将 CANCELED，并且其所有使用者更改为 CASCADE_FAILED。

## 数据节点和先决条件

如果某个数据节点配置作为失败活动的输出，则该数据节点更改为 CASCADE_FAILED 状态。数据节点的故障传播到任何关联的先决条件，这些先决条件会更改为 CANCELED 状态。

## 资源

如果依赖于资源的对象处于 FAILED 状态并且资源本身处于 WAITING_ON_DEPENDENCIES 状态，则资源更改为 FINISHED 状态。

## 重新运行级联失败的对象

默认情况下，重新运行任何活动或数据节点仅重新运行关联的资源。但是，在下列条件下，将管道对象上的 failureAndRerunMode 字段设置为 cascade 允许目标对象上的重新运行命令传播到所有使用者：

- 目标对象的使用者处于 CASCADE_FAILED 状态。
- 目标对象的依赖项没有挂起的重新运行命令。
- 目标对象的依赖项未处于 FAILED、CASCADE_FAILED 或 CANCELED 状态。

如果您尝试重新运行 CASCADE_FAILED 对象并且其任意依赖项为 FAILED、CASCADE_FAILED 或 CANCELED，则重新运行将失败，将对象返回 CASCADE_FAILED 状态。要成功重新运行失败的对象，

您必须在依赖关系链中向上跟踪故障，找到故障的源头，然后再重新运行该对象。在资源上发布重新运行命令时，您还尝试重新运行任何依赖于该资源的对象。

### 级联失败和回填

如果您启用级联故障并让管道创建了多个回填，管道运行时错误可能会导致快速连续创建和删除资源，但未执行有用的工作。在保存管道时，AWS Data Pipeline 尝试使用以下警告消息提醒您有关此状况：*Pipeline_object_name* has 'failureAndRerunMode' field set to 'cascade' and you are about to create a backfill with scheduleStartTime *start_time*. This can result in rapid creation of pipeline objects in case of failures. 出现这种情况的原因是级联故障会快速将下游活动设置为 CASCADE_FAILED 并关闭不再需要的 EMR 集群和 EC2 资源。我们建议您在短时间范围内测试管道，以限制这一状况的影响。

## 管道定义文件语法

此部分中的说明针对使用 AWS Data Pipeline 命令行界面 (CLI) 手动处理管道定义文件。这是使用 AWS Data Pipeline 控制台交互式设计管道的替代方法。

您可以使用任意支持以 UTF-8 文件格式保存文件的文本编辑器手动创建管道定义文件，然后使用 AWS Data Pipeline 命令行界面提交文件。

AWS Data Pipeline 还支持管道定义中的各种复杂的表达式和函数。有关更多信息，请参阅管道表达式和函数。

### 文件结构

管道创建中的第一步是在管道定义文件中编写管道定义对象。以下示例介绍了管道定义文件的一般结构。此文件定义了两个对象，使用“{”和“}”隔离对象，并以逗号分隔。

在以下示例中，第一个对象定义了两个名称/值对，称为字段。第二个对象定义了三个字段。

```
{
  "objects" : [
    {
       "name1" : "value1",
       "name2" : "value2"
    },
    {
       "name1" : "value3",
       "name3" : "value4",
```

```
        "name4" : "value5"
      }
    ]
}
```

创建管道定义文件时，您必须选择需要的管道对象的类型，将其添加到管道定义文件，然后添加适当的字段。有关管道对象的更多信息，请参阅管道对象引用。

例如，您可以为输入数据节点创建一个管道定义对象，为输出数据节点另外创建一个。然后，为活动创建另一个管道定义对象，例如使用 Amazon EMR 处理输入数据。

## 管道字段

在您知道哪些对象类型要包括在管道定义文件中之后，将字段添加到各个管道对象的定义。字段名称括在引号内，并使用空格、冒号、空格与字段值分隔，如下例中所示。

```
"name" : "value"
```

字段值可以是文本字符串、对另一对象的引用、函数调用、表达式或以上任意类型的有序列表。有关可用于字段值的数据类型的更多信息，请参阅简单数据类型。有关您可用于对字段求值的函数的更多信息，请参阅 表达式计算。

字段限制为 2048 个字符。对象大小可以为 20KB，这意味着您不能将过多大字段添加到一个对象。

每个管道对象必须包含以下字段：id 和 type，如以下示例所示。根据对象类型，可能还会需要其他字段。为 id 选择一个对您而言有意义并且在管道定义中唯一的值。type 的值指定对象的类型。指定一个支持的管道定义对象类型，该类型在主题管道对象引用中列出。

```
{
  "id": "MyCopyToS3",
  "type": "CopyActivity"
}
```

有关各个对象的必需和可选字段的更多信息，请参阅对象的文档。

要将来自一个对象的字段包括到其他对象中，请使用 parent 字段并引用对象。例如，对象“B”包含其字段“B1”和“B2”，以及来自对象“A”、“A1”和“A2”的字段。

```
{
```

```
  "id" : "A",
  "A1" : "value",
  "A2" : "value"
},
{
  "id" : "B",
  "parent" : {"ref" : "A"},
  "B1" : "value",
  "B2" : "value"
}
```

您可以使用 ID“Default”定义对象中的常见字段。这些字段自动包括在管道定义文件中的每个对象中，并不明确设置其 parent 字段来引用不同对象。

```
{
  "id" : "Default",
  "onFail" : {"ref" : "FailureNotification"},
  "maximumRetries" : "3",
  "workerGroup" : "myWorkerGroup"
}
```

## 用户定义字段

您可以在管道组件上创建用户定义字段或自定义字段，并通过表达式来引用它们。下面的示例显示了一个名为 myCustomField 的自定义字段，并将 my_customFieldReference 添加到 S3DataNode 对象：

```
{
  "id": "S3DataInput",
  "type": "S3DataNode",
  "schedule": {"ref": "TheSchedule"},
  "filePath": "s3://bucket_name",
  "myCustomField": "This is a custom value in a custom field.",
  "my_customFieldReference": {"ref":"AnotherPipelineComponent"}
  },
```

用户定义字段必须将全小写的单词“my”作为名称前缀，后跟大写字母或下划线。此外，用户定义的字段可以为字符串值，例如前面的 myCustomField 示例，或者引用另一个管道组件，例如前面的 my_customFieldReference 示例。

> **Note**
>
> 在用户定义字段上，AWS Data Pipeline 仅检查对其他管道组件的有效引用，而不检查您添加的任意自定义字段字符串值。

# 使用 API

> **Note**
>
> 如果您不编写与 AWS Data Pipeline 交互的程序，则无需安装任意 AWS 开发工具包。您可以使用控制台或命令行界面创建和运行管道。有关更多信息，请参阅对 AWS Data Pipeline 进行设置。

编写与 AWS Data Pipeline 交互的应用程序或实施自定义任务运行程序的最简单方法是使用 AWS 开发工具包。AWS 开发工具包提供了一项功能，可简化从您的首选编程环境调用 Web 服务 API 的过程。有关更多信息，请参阅安装 AWS 开发工具包。

## 安装 AWS 开发工具包

AWS SDK 提供的函数包装 API 并关注许多连接详细信息，如计算签名、处理请求重试和错误处理。软件开发工具包也包含示例代码、教程和其它资源，以便帮助您开始编写调用 AWS 的应用程序。调用软件开发工具包中的包装函数能大大简化编写 AWS 应用程序的过程。有关如何下载和使用 AWS 开发工具包的详细信息，请转到示例代码和库。

AWS Data Pipeline 的开发工具包支持可用于以下平台：

- 适用于 Java 的 AWS 软件开发工具包
- 适用于 Node.js 的 AWS 开发工具包
- 适用于 PHP 的 AWS 开发工具包
- 适用于 Python 的 AWS 开发工具包 (Boto)
- 适用于 Ruby 的 AWS 开发工具包
- 适用于 .NET 的 Amazon SDK

# 向 AWS Data Pipeline 发出 HTTP 请求

有关 AWS Data Pipeline 中编程对象的完整说明，请参阅 AWS Data Pipeline API 参考。

如果您没有使用任何 AWS 开发工具包，可以使用 POST 请求方法通过 HTTP 执行 AWS Data Pipeline 操作。POST 方法需要您在请求标头中指定具体操作，并在请求正文中以 JSON 格式提供操作数据。

## HTTP 标头内容

AWS Data Pipeline 要求在 HTTP 请求标头中包含以下信息：

- `host`：AWS Data Pipeline 终端节点。

  有关终端节点的信息，请参阅区域和终端节点。
- `x-amz-date` 您必须在 HTTP Date 标头或 AWS x-amz-date 标头中提供时间戳。(有些 HTTP 客户端库不允许设置 Date 标头。) 当 x-amz-date 标头存在时，系统会在请求身份验证期间忽略所有 Date 标头。

  必须利用以下三种格式中的一种来指定数据，如 HTTP/1.1 RFC 中所规定：
  - 格林威治时间 1994 年 11 月 6 日，星期日 08:49:37（RFC 822，由 RFC 1123 更新）
  - 格林威治时间 1994 年 11 月 6 日，星期日 08:49:37（RFC 850，由 RFC 1036 废弃）
  - 1994 年 11 月 6 日 08:49:37，星期六 (ANSI C asctime() 格式)
- `Authorization`：授权参数的集合，AWS 使用这些参数确保请求的有效性和可靠性。有关配置标头的更多信息，请转至 Signature Version 4 Signing Process。
- `x-amz-target` 请求的目标服务和数据的操作，格式如下：`<<serviceName>>_<<API version>>.<<operationName>>`

  例如 `DataPipeline_20121129.ActivatePipeline`
- `content-type`：指定 JSON 和版本。例如 `Content-Type: application/x-amz-json-1.0`

以下示例为激活管道所用的 HTTP 请求的标头。

```
POST / HTTP/1.1
host: https://datapipeline.us-east-1.amazonaws.com
x-amz-date: Mon, 12 Nov 2012 17:49:52 GMT
```

```
x-amz-target: DataPipeline_20121129.ActivatePipeline
Authorization: AuthParams
Content-Type: application/x-amz-json-1.1
Content-Length: 39
Connection: Keep-Alive
```

## HTTP 正文内容

HTTP 请求的正文包含 HTTP 请求标头中指定的操作数据。数据的格式必须遵照每个 AWS Data Pipeline API 的 JSON 数据架构。AWS Data Pipeline JSON 数据架构定义了每个操作可用的数据和参数（例如比较运算符和枚举常量）的类型。

确定 HTTP 请求正文的格式

使用 JSON 数据格式可以同时传递数据值和数据结构。以方括号注释的形式可以在元素中嵌套其他元素。以下示例显示了放置由三个对象及其对应槽组成的管道定义的请求。

```
{
 "pipelineId": "df-00627471SOVYZEXAMPLE",
 "pipelineObjects":
  [
    {"id": "Default",
     "name": "Default",
     "slots":
      [
        {"key": "workerGroup",
         "stringValue": "MyWorkerGroup"}
      ]
    },
    {"id": "Schedule",
     "name": "Schedule",
     "slots":
      [
       {"key": "startDateTime",
         "stringValue": "2012-09-25T17:00:00"},
       {"key": "type",
         "stringValue": "Schedule"},
       {"key": "period",
         "stringValue": "1 hour"},
       {"key": "endDateTime",
         "stringValue": "2012-09-25T18:00:00"}
      ]
```

```
    },
    {"id": "SayHello",
     "name": "SayHello",
     "slots":
      [
        {"key": "type",
         "stringValue": "ShellCommandActivity"},
        {"key": "command",
         "stringValue": "echo hello"},
        {"key": "parent",
         "refValue": "Default"},
        {"key": "schedule",
         "refValue": "Schedule"}

      ]
    }
  ]
}
```

处理 HTTP 响应

以下是 HTTP 响应中的一些重要标头，以及您在应用程序中对其进行处理的方法：

- HTTP/1.1 - 此标头后跟状态代码，200 的代码值表示操作成功。任何其他值指示错误。
- x-amzn-RequestId - 此标头包含请求 ID，如果需要使用 AWS Data Pipeline 对请求进行问题排查，就可以使用此 ID。请求 ID 的示例：K2QH8DNOU907N97FNA2GDLL8OBVV4KQNSO5AEMVJF66Q9ASUAAJG。
- x-amz-crc32 - AWS Data Pipeline 计算 HTTP 负载的 CRC32 校验和，并在 x-amz-crc32 标头中返回此校验和。我们建议您在客户端计算自己的 CRC32 校验和，并将其与 x-amz-crc32 标头进行比较。如果校验和不匹配，则可能表示传输中发生数据损坏。如果发生数据毁坏，则应重试请求。

AWS 开发工具包用户不需要手动执行这种验证，因为开发工具包会从 Amazon DynamoDB 中计算每个回复的校验和，如果检测到不匹配就会自动重试。

AWS Data Pipeline JSON 请求和响应示例

以下示例显示了用于创建新管道的请求。然后，它显示了 AWS Data Pipeline 响应，包括新创建管道的管道标识符。

### HTTP POST 请求

```
POST / HTTP/1.1
host: https://datapipeline.us-east-1.amazonaws.com
x-amz-date: Mon, 12 Nov 2012 17:49:52 GMT
x-amz-target: DataPipeline_20121129.CreatePipeline
Authorization: AuthParams
Content-Type: application/x-amz-json-1.1
Content-Length: 50
Connection: Keep-Alive

{"name": "MyPipeline",
 "uniqueId": "12345ABCDEFG"}
```

### AWS Data Pipeline 响应

```
HTTP/1.1 200
x-amzn-RequestId: b16911ce-0774-11e2-af6f-6bc7a6be60d9
x-amz-crc32: 2215946753
Content-Type: application/x-amz-json-1.0
Content-Length: 2
Date: Mon, 16 Jan 2012 17:50:53 GMT

{"pipelineId": "df-00627471SOVYZEXAMPLE"}
```

# AWS Data Pipeline 中的安全性

AWS 十分重视云安全性。为了满足对安全性最敏感的组织的需求，我们打造了具有超高安全性的数据中心和网络架构。作为 AWS 客户，您也将从这些数据中心和网络架构受益。

安全性是 AWS 和您的共同责任。责任共担模型将其描述为云的安全性和云中的安全性：

- 云的安全性 – AWS 负责保护在 AWS 云中运行 AWS 服务的基础设施。AWS 还向您提供可安全使用的服务。第三方审核员定期测试和验证我们的安全性的有效性，作为 AWS Compliance Programs 的一部分。要了解适用于 AWS Data Pipeline 的合规性计划，请参阅合规性计划范围内的 AWS 服务。
- 云中的安全性 - 您的责任由您使用的 AWS 服务决定。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。

此文档将帮助您了解如何在使用 AWS Data Pipeline 时应用责任共担模型 以下主题说明如何配置 AWS Data Pipeline 以实现您的安全性和合规性目标。您还将了解如何使用其他亚马逊云科技服务来帮助您监控和保护 AWS Data Pipeline 资源。

主题



## AWS Data Pipeline 中的数据保护

AWS责任共担模式适用于 AWS Data Pipeline 中的数据保护。如该模式中所述，AWS 负责保护运行所有 AWS Cloud 的全球基础设施。您负责维护对托管在此基础设施上的内容的控制。此内容包括您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。有关欧洲数据保护的信息，请参阅 AWS 安全性博客 上的 AWS 责任共担模式和 GDPR 博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置单个用户。这样，每个用户只获得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您的数据：

- 对每个账户使用 multi-factor authentication（MFA）。
- 使用 SSL/TLS 与 AWS 资源进行通信。建议使用 TLS 1.2 或更高版本。
- 使用 AWS CloudTrail 设置 API 和用户活动日志记录。
- 使用 AWS 加密解决方案以及 AWS 服务 中的所有默认安全控制。
- 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 中的敏感数据。
- 如果在通过命令行界面或 API 访问 AWS 时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准 (FIPS) 第 140-2 版》。
- AWS Data Pipeline 支持 Amazon EMR 的 IMDSv2 以及 Amazon EC2 资源。要将 IMDSv2 与 Amazon EMR 一起使用，请使用 5.23.1、5.27.1、5.32 或更高版本，或 6.2 或更高版本。有关更多信息，请参阅配置到 Amazon EC2 实例的元数据服务请求和使用 IMDSv2。

我们强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如 Name（名称）字段）。这包括使用控制台、API、AWS CLI 或 AWS SDK 处理 AWS Data Pipeline 或其他 AWS 服务时。您在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供 URL，我们强烈建议您不要在 URL 中包含凭证信息来验证您对该服务器的请求。

# 适用于 AWS Data Pipeline 的 Identity and Access Management

您的安全凭证使 AWS 中的服务可以识别您，并授予您对 AWS 资源（例如您的管道）的使用权限。利用 AWS Data Pipeline 和 AWS Identity and Access Management (IAM) 功能，可在不共享您的安全凭证的情况下允许 AWS Data Pipeline 和其他用户访问您的 AWS Data Pipeline 资源。

组织可以共享对管道的访问权限，这样该组织中的个人可以合作开发和维护。不过，可能需要执行以下操作：

- 控制哪些用户可以访问特定的管道
- 保护生产管道，避免被错误编辑
- 允许审计员具有管道的只读访问权限，但阻止他们进行更改

AWS Data Pipeline 与 AWS Identity and Access Management (IAM) 集成，后者提供多种功能：

- 在 AWS 账户 下创建用户和组
- 在您的 AWS 账户 中的用户之间轻松共享您的 AWS 资源。
- 为每个用户分配具有唯一性的安全凭证
- 控制每个用户对服务和资源的访问
- 为您的 AWS 账户 中的所有用户获取统一账单。

通过将 IAM 与 AWS Data Pipeline 配合使用，您可以控制组织中的用户能否使用特定的 API 操作执行任务，以及他们能否使用特定的 AWS 资源。您可以基于管道标签和工作线程组使用 IAM 策略，将您的管道与其他用户分享并控制这些用户拥有的访问级别。

目录



# 适用于 AWS Data Pipeline 的 IAM 策略

默认情况下，IAM 实体没有创建或修改 AWS 资源的权限。要允许 IAM 实体创建或修改资源和执行任务，您必须创建 IAM policy 以允许 IAM 实体使用他们所需的特定资源和 API 操作，然后将这些策略与需要这些权限的 IAM 实体关联起来。

在将策略附加到一个用户或一组用户时，它会授权或拒绝用户使用指定资源执行指定任务。有关 IAM policy 的一般信息，请参阅 IAM 用户指南中的权限与策略。有关管理和创建自定义 IAM policy 的更多信息，请参阅管理 IAM policy。

目录



## 策略语法

IAM policy 是包含一个或多个语句的 JSON 文档。每个语句的结构如下：

```
{
```

```
  "Statement":[{
    "Effect":"effect",
    "Action":"action",
    "Resource":"*",
    "Condition":{
      "condition":{
        "key":"value"
        }
      }
    }
  ]
}
```

策略声明包含以下元素：

- Effect：此 effect 可以是 Allow 或 Deny。在默认情况下，IAM 实体没有使用资源和 API 操作的许可，因此，所有请求均会被拒绝。显式允许将覆盖默认规则。显式拒绝将覆盖任何允许。
- Action：action 是对其授予或拒绝权限的特定 API 操作。要查看 AWS Data Pipeline 操作的列表，请参阅 AWS Data Pipeline API 参考中的操作。
- Resource：受操作影响的资源。此处唯一有效值为 "*"。
- Condition：条件是可选的。它们可以用于控制策略生效的时间。

  AWS Data Pipeline 会实施 AWS 范围内的上下文键（请参阅可用的条件键）以及以下服务特定的键。

  - datapipeline:PipelineCreator - 向创建管道的用户授予访问权限。有关示例，请参阅授予管道所有者完全访问权限。
  - datapipeline:Tag - 基于管道标记授予访问权限。有关更多信息，请参阅使用标签控制对管道的访问权限。
  - datapipeline:workerGroup - 基于工作线程组的名称授予访问权限。有关更多信息，请参阅使用工作线程组控制对管道的访问权限。

## 使用标签控制对管道的访问权限

您可以创建引用管道标签的 IAM 策略。这使您能够使用管道标签来执行以下操作：

- 授予对管道的只读访问权限
- 授予对管道的读/写访问权限
- 阻止对管道的访问

例如，假设一个管理员有生产和开发两个管道环境，每个环境有一个 IAM 组。对于生产环境中的管道，管理员向生产 IAM 组中的用户授予读/写访问权限，但向开发人员 IAM 组中的用户授予只读访问权限。对于开发环境中的管道，管理员向生产和开发人员 IAM 组中的用户均授予读/写访问权限。

要实现这一场景，管理员使用“environment = production”标签来标记生产管道，并将以下策略附加到开发人员 IAM 组。第一条语句授予对所有管道的只读访问权限。第二条语句授予对没有“environment = production”标签的管道的读/写访问权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "datapipeline:Describe*",
        "datapipeline:ListPipelines",
        "datapipeline:GetPipelineDefinition",
        "datapipeline:QueryObjects"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "datapipeline:*",
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {"datapipeline:Tag/environment": "production"}
      }
    }
  ]
}
```

此外，管理员将以下策略附加到生产 IAM 组。此语句授予对所有管道的完全访问权限。

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

```
  ]
}
```

有关更多示例，请参阅基于标签向用户授予只读访问权限和基于标签向用户授予完全访问权限。

## 使用工作线程组控制对管道的访问权限

您可以创建引用工作线程组名称的 IAM 策略。

例如，假设一个管理员有生产和开发两个管道环境，每个环境有一个 IAM 组。管理员有三个数据库服务器，分别为生产、生产前和开发人员环境配置了任务运行程序。管理员想要确保用户在生产 IAM 组中的用户可以创建推送任务到生产资源的管道，而开发 IAM 组中的用户可以创建推送任务到生产前和开发人员资源的管道。

要实现这种场景，管理员使用生产凭证在生产资源上安装任务运行程序，并将 `workerGroup` 设置为“prodresource”。此外，管理员使用开发凭证在开发资源上安装任务运行程序，并将 `workerGroup` 设置为“pre-production”和“development”。管理员将以下策略附加到开发人员 IAM 组来阻止对“prodresource”资源的访问。第一条语句授予对所有管道的只读访问权限。第二条语句在工作线程组的名称具有前缀“dev”或“pre-prod”时，授予对管道的读/写访问权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "datapipeline:Describe*",
        "datapipeline:ListPipelines",
        "datapipeline:GetPipelineDefinition",
        "datapipeline:QueryObjects"
      ],
      "Resource": "*"
    },
    {
      "Action": "datapipeline:*",
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "datapipeline:workerGroup": ["dev*","pre-prod*"]
        }
      }
```

```
    }
  ]
}
```

此外，管理员将以下策略附加到生产 IAM 组来授予对“prodresource”资源的访问权限。第一条语句授予对所有管道的只读访问权限。第二条语句在工作线程组的名称具有前缀“prod”时，授予读/写访问权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "datapipeline:Describe*",
        "datapipeline:ListPipelines",
        "datapipeline:GetPipelineDefinition",
        "datapipeline:QueryObjects"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "datapipeline:*",
      "Resource": "*",
      "Condition": {
        "StringLike": {"datapipeline:workerGroup": "prodresource*"}
      }
    }
  ]
}
```

## AWS Data Pipeline 策略示例

以下示例演示如何授予用户对管道的完全或受限访问权限。

目录

## 示例 1：基于标签授予用户只读访问权限

以下策略允许用户使用只读 AWS Data Pipeline API 操作，但仅限于具有标签“environment = production”的管道。

ListPipelines API 操作不支持基于标签的授权。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "datapipeline:Describe*",
        "datapipeline:GetPipelineDefinition",
        "datapipeline:ValidatePipelineDefinition",
        "datapipeline:QueryObjects"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "datapipeline:Tag/environment": "production"
        }
      }
    }
  ]
}
```

## 示例 2：基于标签授予用户完全访问权限

以下策略允许用户使用所有 AWS Data Pipeline API 操作（ListPipelines 例外），但仅限于具有标签“environment = test”的管道。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "datapipeline:*"
      ],
```

```
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "datapipeline:Tag/environment": "test"
        }
      }
    }
  ]
}
```

## 示例 3：授予管道所有者完全访问权限

以下策略允许用户使用所有 AWS Data Pipeline API 操作，但仅限其自己的管道。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "datapipeline:*"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "datapipeline:PipelineCreator": "${aws:userid}"
        }
      }
    }
  ]
}
```

## 示例 4：授予用户对 AWS Data Pipeline 控制台的访问权限

以下策略允许用户使用 AWS Data Pipeline 控制台创建和管理管道。

此策略包含 `PassRole` 权限的操作，该权限用于 AWS Data Pipeline 需要的 `roleARN` 所关联的特定资源。有关基于身份的 (IAM) `PassRole` 权限的更多信息，请参阅博文授予权限，以启动具有 IAM 角色的 EC2 实例（PassRole 权限）。

```
{
 "Version": "2012-10-17",
 "Statement": [{
   "Action": [
    "cloudwatch:*",
    "datapipeline:*",
    "dynamodb:DescribeTable",
    "elasticmapreduce:AddJobFlowSteps",
    "elasticmapreduce:ListInstance*",
    "iam:AddRoleToInstanceProfile",
    "iam:CreateInstanceProfile",
    "iam:GetInstanceProfile",
    "iam:GetRole",
    "iam:GetRolePolicy",
    "iam:ListInstanceProfiles",
    "iam:ListInstanceProfilesForRole",
    "iam:ListRoles",
    "rds:DescribeDBInstances",
    "rds:DescribeDBSecurityGroups",
    "redshift:DescribeClusters",
    "redshift:DescribeClusterSecurityGroups",
    "s3:List*",
    "sns:ListTopics"
   ],
   "Effect": "Allow",
   "Resource": [
    "*"
   ]
  },
  {
   "Action": "iam:PassRole",
   "Effect": "Allow",
   "Resource": [
    "arn:aws:iam::*:role/DataPipelineDefaultResourceRole",
    "arn:aws:iam::*:role/DataPipelineDefaultRole"
   ]
  }
 ]
}
```

# 适用于 AWS Data Pipeline 的 IAM 角色

AWS Data Pipeline 使用 AWS Identity and Access Management 角色。附加到 IAM 角色的权限策略决定了您的应用程序可以执行哪些操作 AWS Data Pipeline 以及它们可以访问哪些 AWS 资源。有关更多信息，请参阅《IAM 用户指南》中的 IAM 角色。

AWS Data Pipeline 需要两个 IAM 角色：

- 管道角色控制 AWS Data Pipeline 对 AWS 资源的访问。在管道对象定义中，`role` 字段指定此角色。
- EC2 实例角色控制在 EC2 实例上运行的应用程序（包括 Amazon EMR 集群中的 EC2 实例）对 AWS 资源的访问权限。在管道对象定义中，`resourceRole` 字段指定此角色。

> **Important**
>
> 如果您在 2022 年 10 月 3 日之前使用带有默认角色的 AWS Data Pipeline 控制台，AWS Data Pipeline 为您创建了 `DataPipelineDefaultRole` 并将 `AWSDataPipelineRole` 托管策略附加到该角色。自 2022 年 10 月 3 日起，`AWSDataPipelineRole` 托管策略已被弃用，使用控制台时必须为管道指定管道角色。
>
> 我们建议您查看现有管道，并确定 `DataPipelineDefaultRole` 是否与管道相关联以及 `AWSDataPipelineRole` 是否已关联到该角色。如果是，请查看此策略允许的访问权限，以确保其符合您的安全要求。根据需要添加、更新或替换附加到此角色的策略和策略语句。或者，您也可以更新管道以使用您创建的具有不同权限策略的角色。

## AWS Data Pipeline 角色权限策略示例

每个角色都附加了一个或多个权限策略，用于确定该角色可以访问的 AWS 资源以及该角色可以执行的操作。本主题提供了管道角色的权限策略示例。它还提供了 `AmazonEC2RoleforDataPipelineRole`（默认 EC2 实例角色的托管策略 `DataPipelineDefaultResourceRole`）的内容。

管道角色权限策略示例

以下示例策略的范围经过限定，允许 AWS Data Pipeline 需要使用 Amazon EC2 和 Amazon EMR 资源运行管道的基本功能。它还提供了许多管道所要求的访问其他 AWS 资源所需的权限，如 Amazon Simple Storage Service 和 Amazon Simple Notification Service。如果管道中定义的对象不

需要 AWS 服务的资源，我们强烈建议您移除用于访问该服务的权限。例如，如果您的管道未定义 DynamoDBDataNode 或未使用 SnsAlarm 操作，我们建议您删除这些操作的允许语句。

- 将 *111122223333* 替换为您的AWS账户 ID。
- *NameOfDataPipelineRole* 替换为管道角色的名称（此策略所关联的角色）。
- *NameOfDataPipelineResourceRole* 替换为 EC2 实例角色的名称。
- *us-west-1* 替换为适合您的应用程序的区域。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetInstanceProfile",
                "iam:GetRole",
                "iam:GetRolePolicy",
                "iam:ListAttachedRolePolicies",
                "iam:ListRolePolicies",
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::111122223333:role/NameOfDataPipelineRole",
                "arn:aws:iam::111122223333 :role/NameOfDataPipelineResourceRole"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CancelSpotInstanceRequests",
                "ec2:CreateNetworkInterface",
                "ec2:CreateSecurityGroup",
                "ec2:CreateTags",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteSecurityGroup",
                "ec2:DeleteTags",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeDhcpOptions",
```

```
                "ec2:DescribeImages",
                "ec2:DescribeInstanceStatus",
                "ec2:DescribeInstances",
                "ec2:DescribeKeyPairs",
                "ec2:DescribeLaunchTemplates",
                "ec2:DescribeNetworkAcls",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribePrefixLists",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSpotInstanceRequests",
                "ec2:DescribeSpotPriceHistory",
                "ec2:DescribeSubnets",
                "ec2:DescribeTags",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeVpcEndpointServices",
                "ec2:DescribeVpcs",
                "ec2:DetachNetworkInterface",
                "ec2:ModifyImageAttribute",
                "ec2:ModifyInstanceAttribute",
                "ec2:RequestSpotInstances",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:RunInstances",
                "ec2:TerminateInstances",
                "ec2:DescribeVolumeStatus",
                "ec2:DescribeVolumes",
                "elasticmapreduce:TerminateJobFlows",
                "elasticmapreduce:ListSteps",
                "elasticmapreduce:ListClusters",
                "elasticmapreduce:RunJobFlow",
                "elasticmapreduce:DescribeCluster",
                "elasticmapreduce:AddTags",
                "elasticmapreduce:RemoveTags",
                "elasticmapreduce:ListInstanceGroups",
                "elasticmapreduce:ModifyInstanceGroups",
                "elasticmapreduce:GetCluster",
                "elasticmapreduce:DescribeStep",
                "elasticmapreduce:AddJobFlowSteps",
                "elasticmapreduce:ListInstances",
                "iam:ListInstanceProfiles",
                "redshift:DescribeClusters"
            ],
            "Resource": [
```

```
            "*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "sns:GetTopicAttributes",
            "sns:Publish"
        ],
        "Resource": [
            "arn:aws:sns:us-west-1:111122223333:MyFirstSNSTopic",
            "arn:aws:sns:us-west-1:111122223333:MySecondSNSTopic",
            "arn:aws:sns:us-west-1:111122223333:AnotherSNSTopic"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:ListBucket",
            "s3:ListMultipartUploads"
        ],
        "Resource": [
          "arn:aws:s3:::MyStagingS3Bucket",
          "arn:aws:s3:::MyLogsS3Bucket",
          "arn:aws:s3:::MyInputS3Bucket",
          "arn:aws:s3:::MyOutputS3Bucket",
          "arn:aws:s3:::AnotherRequiredS3Buckets"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:GetObject",
            "s3:GetObjectMetadata",
            "s3:PutObject"
        ],
        "Resource": [
            "arn:aws:s3:::MyStagingS3Bucket/*",
            "arn:aws:s3:::MyLogsS3Bucket/*",
            "arn:aws:s3:::MyInputS3Bucket/*",
            "arn:aws:s3:::MyOutputS3Bucket/*",
            "arn:aws:s3:::AnotherRequiredS3Buckets/*"
        ]
    },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:Scan",
                "dynamodb:DescribeTable"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-west-1:111122223333:table/MyFirstDynamoDBTable",
                "arn:aws:dynamodb:us-west-1:111122223333:table/MySecondDynamoDBTable",
                "arn:aws:dynamodb:us-west-1:111122223333:table/AnotherDynamoDBTable"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "rds:DescribeDBInstances"
            ],
            "Resource": [
                "arn:aws:rds:us-west-1:111122223333:db:MyFirstRdsDb",
                "arn:aws:rds:us-west-1:111122223333:db:MySecondRdsDb",
                "arn:aws:rds:us-west-1:111122223333:db:AnotherRdsDb"
            ]
        }
    ]
}
```

### EC2 实例角色的默认托管策略

下面显示的是 AmazonEC2RoleforDataPipelineRole 的内容。这是附加到 DataPipelineDefaultResourceRole AWS Data Pipeline 的默认资源角色的托管策略。在为管道定义资源角色时，我们建议您从此权限策略开始，然后删除不需要的 AWS 服务操作的权限。

所示的是该策略的第 3 版，这在撰写本文时是最新版本。在 IAM 控制台中查看最新版的策略。

```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Effect": "Allow",
      "Action": [
        "cloudwatch:*",
        "datapipeline:*",
        "dynamodb:*",
        "ec2:Describe*",
```

```
        "elasticmapreduce:AddJobFlowSteps",
        "elasticmapreduce:Describe*",
        "elasticmapreduce:ListInstance*",
        "elasticmapreduce:ModifyInstanceGroups",
        "rds:Describe*",
        "redshift:DescribeClusters",
        "redshift:DescribeClusterSecurityGroups",
        "s3:*",
        "sdb:*",
        "sns:*",
        "sqs:*"
      ],
      "Resource": ["*"]
    }]
}
```

## 为 AWS Data Pipeline 创建 IAM 角色和编辑角色权限

在 IAM 控制台中，请按照以下过程为 AWS Data Pipeline 创建角色。该过程包括两个步骤。首先，创建一个权限策略以附加到该角色。然后，创建一个角色并附加到该策略。创建角色后，您可以通过附加和分离权限策略来更改角色的权限。

> **Note**
>
> 当您按照下文所述使用控制台为 AWS Data Pipeline 创建角色时，IAM 会创建并附加该角色所需的相应信任策略。

创建权限策略以与 AWS Data Pipeline 角色配合使用

1. 访问：https://console.aws.amazon.com/iam/，打开 IAM 控制台。
2. 在导航窗格中，选择 Policies (策略)，然后选择 Create policy (创建策略)。
3. 请选择 JSON 选项卡。
4. 如果您要创建管道角色，请将策略示例的内容复制并粘贴到 管道角色权限策略示例 中，然后根据您的安全要求酌情对其进行编辑。或者，如果您要创建自定义 EC2 实例角色，请对 EC2 实例角色的默认托管策略 中的示例执行相同的操作。
5. 选择Review policy（查看策略）。
6. 输入您策略的名称（如 `MyDataPipelineRolePolicy`）和可选描述，然后选择创建策略。

7. 记下策略的名称。当您创建角色时，您将需要它。

为 AWS Data Pipeline 创建 IAM 角色

1. 访问：https://console.aws.amazon.com/iam/，打开 IAM 控制台。
2. 在导航窗格中，选择角色，然后选择创建角色。
3. 在选择使用案例下，选择数据管道。
4. 在选择您的使用案例下，请执行以下操作之一：
   - 选择 `Data Pipeline` 以创建管道角色。
   - 选择 `EC2 Role for Data Pipeline` 以创建资源角色。
5. 选择Next: Permissions（下一步: 权限）。
6. 如果列出了 AWS Data Pipeline 的默认策略，请继续执行以下步骤来创建角色，然后根据下一个过程中的说明对其进行编辑。否则，输入您在上一个过程中创建的策略的名称，然后从列表中选择它。
7. 选择下一步：标签，输入任何要添加到角色的标签，然后选择下一步：审核。
8. 输入角色的名称（如 `MyDataPipelineRole`）和可选描述，然后选择创建角色。

为 AWS Data Pipeline 的 IAM 角色附加或取消附加权限策略

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择角色
3. 在搜索框中，开始键入要编辑的角色的名称，例如 DataPipelineDefaultRole 或 MyDataPipelineRole，然后从列表中选择角色名称。
4. 在权限选项卡上，执行以下操作：
   - 要分离权限策略，请在权限策略下，选择策略条目最右侧的“删除”按钮。当系统提示进行确认时，选择分离。
   - 要附加您之前创建的策略，请选择附加策略。在搜索框中，开始键入要编辑的策略的名称，从列表中选中它，然后选择附加策略。

## 更改现有管道的角色

如果要为管道分配不同的管道角色或资源角色，则可以在 AWS Data Pipeline 控制台中使用架构师编辑器。

使用控制台来编辑分配给管道的角色

1. 打开 AWS Data Pipeline 控制台，网址为 https://console.aws.amazon.com/datapipeline/。
2. 从列表中选择管道，然后选择操作、编辑。
3. 在架构师编辑器的右侧窗格中，选择其他。
4. 从资源角色和角色列表中，选择要为 AWS Data Pipeline 分配的角色，然后选择保存。

# AWS Data Pipeline 中的日志记录和监控

AWS Data Pipeline 与 AWS CloudTrail 集成，后者是在 AWS 中记录用户、角色或 AWS Data Pipeline 服务所执行操作的服务。CloudTrail 将 AWS Data Pipeline 的所有 API 调用作为事件捕获。捕获的调用包含来自 AWS Data Pipeline 控制台和代码的 AWS Data Pipeline API 操作调用。如果您创建跟踪，则可以使 CloudTrail 事件持续传送到 Amazon S3 存储桶（包括 AWS Data Pipeline 的事件）。如果您不配置跟踪记录，则仍可在 CloudTrail 控制台中的 Event history（事件历史记录）中查看最新事件。使用 CloudTrail 收集的信息，您可以确定向 AWS Data Pipeline 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

要了解有关 CloudTrail 的更多信息，请参阅《AWS CloudTrail 用户指南》。

## CloudTrail 中的 AWS Data Pipeline 信息

在您创建 AWS 账户时，将在该账户上启用 CloudTrail。当 AWS Data Pipeline 中发生活动时，该活动将记录在 CloudTrail 事件中，并与其他 AWS 服务事件一同保存在 Event history (事件历史记录) 中。您可以在 AWS 账户中查看、搜索和下载最新事件。有关更多信息，请参阅使用 CloudTrail 事件历史记录查看事件。

要持续记录 AWS 账户中的事件（包括 AWS Data Pipeline 的事件），请创建跟踪。通过跟踪记录，CloudTrail 可将日志文件传送至 Amazon S3 存储桶。默认情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有亚马逊云科技区域。此跟踪在 AWS 分区中记录所有区域中的事件，并将日志文件传送至您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取行动。有关更多信息，请参阅下列内容：

- 创建跟踪概览
- CloudTrail 支持的服务和集成
- 为 CloudTrail 配置 Amazon SNS 通知
- 从多个区域接收 CloudTrail 日志文件和从多个账户接收 CloudTrail 日志文件

CloudTrail 记录所有 AWS Data Pipeline 操作，且“AWS Data Pipeline API 参考：操作”一章中也介绍了这些操作。例如，对 CreatePipeline 操作的调用会在 CloudTrail 日志文件中生成条目。

每个事件或日志条目都包含有关生成请求的人员信息。身份信息可帮助您确定以下内容：

- 使用根凭证还是 IAM 角色凭证发出请求
- 请求是使用角色还是联合身份用户的临时安全凭证发出的。
- 请求是否由其它 AWS 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

## 了解 AWS Data Pipeline 日志文件条目

跟踪是一种配置，可用于将事件作为日志文件传送到您指定的 Amazon S3 桶。CloudTrail 日志文件包含一个或多个日志条目。一个事件表示来自任何源的一个请求，包括有关所请求的操作、操作的日期和时间、请求参数等方面的信息。CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序显示。

下面的示例显示了一个 CloudTrail 日志条目，该条目说明了 `CreatePipeline` 操作。

```
    {
"Records": [
  {
    "eventVersion": "1.02",
    "userIdentity": {
      "type": "Root",
      "principalId": "123456789012",
      "arn": "arn:aws:iam::aws-account-id:role/role-name",
      "accountId": "role-account-id",
      "accessKeyId": "role-access-key"
    },
    "eventTime": "2014-11-13T19:15:15Z",
    "eventSource": "datapipeline.amazonaws.com",
    "eventName": "CreatePipeline",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "72.21.196.64",
    "userAgent": "aws-cli/1.5.2 Python/2.7.5 Darwin/13.4.0",
    "requestParameters": {
      "name": "testpipeline",
      "uniqueId": "sounique"
```

```
      },
      "responseElements": {
        "pipelineId": "df-06372391ZG65EXAMPLE"
      },
      "requestID": "65cbf1e8-6b69-11e4-8816-cfcbadd04c45",
      "eventID": "9f99dce0-0864-49a0-bffa-f72287197758",
      "eventType": "AwsApiCall",
      "recipientAccountId": "role-account-id"
    },
      ...additional entries
  ]
}
```

# AWS Data Pipeline 中的事件响应

AWS Data Pipeline 的事件响应是一项 AWS 责任。AWS 拥有正式的、已归档的策略和程序来管理事件响应。

具有广泛影响的亚马逊云科技 操作性问题将在 Amazon Service Health Dashboard 上发布。操作性问题也会通过 Personal Health Dashboard 发布给个人账户。

# AWS Data Pipeline 的合规性验证

AWS Data Pipeline 不在任何 AWS 合规性计划范围内。有关特定合规性计划范围内的 Amazon Web Services 的列表，请参阅合规性计划范围内的 AWS 服务。有关一般信息，请参阅 AWS 合规性计划。

# AWS Data Pipeline 中的故障恢复能力

AWS全球基础设施围绕AWS区域和可用区构建。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故障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 区域和可用区的更多信息，请参阅 AWS 全球基础设施。

# AWS Data Pipeline 中的基础设施安全性

作为一项托管式服务，AWS Data Pipeline 由 Amazon Web Services：安全流程概览白皮书中所述的 AWS 全球网络安全程序提供保护。

您可以使用 AWS 发布的 API 调用通过网络访问 AWS Data Pipeline。客户端必须支持传输层安全性 (TLS) 1.0 或更高版本。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve 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 委托人关联的秘密访问密钥来对请求进行签名。或者，您可以使用 AWS Security Token Service (AWS STS) 生成临时安全凭证来对请求进行签名。

# AWS Data Pipeline 中的配置和漏洞分析

配置和 IT 控制是AWS和您（我们的客户）之间的共同责任。有关更多信息，请参阅AWS责任共担模型。

# 教程

以下教程将指导您逐步完成通过 AWS Data Pipeline 创建和使用管道的过程。

教程

- 将 Amazon EMR 与 Hadoop 流式处理结合使用来处理数据
- 使用 AWS Data Pipeline 在 Amazon S3 存储桶之间复制 CSV 数据
- 使用 AWS Data Pipeline 将 MySQL 数据导出到 Amazon S3。
- 使用 AWS Data Pipeline 复制数据到 Amazon Redshift

# 将 Amazon EMR 与 Hadoop 流式处理结合使用来处理数据

您可以使用 AWS Data Pipeline 管理您的 Amazon EMR 集群。利用 AWS Data Pipeline，您可以指定先决条件（必须先满足该先决条件，然后才能启动集群；例如，确保将今天的数据上传到 Amazon S3）、重复运行集群的计划以及要使用的集群配置。以下教程将引导您完成启动简单集群的过程。

在本教程中，您将为简单 Amazon EMR 集群创建一个管道来运行由 Amazon EMR 提供的预先存在的 Hadoop 流式处理作业，并在任务成功完成后发送 Amazon SNS 通知。您为此任务使用由 AWS Data Pipeline 提供的 Amazon EMR 集群资源。该示例应用程序称作 WordCount，也可从 Amazon EMR 控制台手动运行它。请注意，由 AWS Data Pipeline 代表您生成的集群将显示在 Amazon EMR 控制台中并对您的 Amazon Web Services account 计费。

管道对象

管道使用以下对象：

EmrActivity

定义要在管道中执行的工作（运行由 Amazon EMR 提供的预先存在的 Hadoop 流式处理作业）。

EmrCluster

AWS Data Pipeline 用来执行此活动的资源。

集群是一组 Amazon EC2 实例。AWS Data Pipeline 启动集群，然后在任务完成后终止集群。

计划

此活动的开始日期、时间和持续时间。您可以选择指定结束日期和时间。

SnsAlarm

在任务成功完成后，向您指定的主题发送 Amazon SNS 通知。

目录



## 开始前的准备工作

确保您已完成以下步骤。

- 完成对 AWS Data Pipeline 进行设置中所述的任务。
- (可选) 为集群设置一个 VPC，并为该 VPC 设置一个安全组。
- 创建用于发送电子邮件通知的主题并记下主题的 Amazon 资源名称 (ARN)。有关更多信息，请参阅《Amazon Simple Notification Service 入门指南》 中的创建主题。

## 使用命令行启动集群

如果您定期运行 Amazon EMR 集群来分析 Web 日志或科学数据，则可以使用 AWS Data Pipeline 管理您的 Amazon EMR 集群。利用 AWS Data Pipeline，您可以指定先决条件（必须先满足这些先决条件，然后才能启动集群；例如，确保将今天的数据上传到 Amazon S3。）本教程将引导您完成启动集群的过程，该集群可以是基于 Amazon EMR 的简单管道的模型，也可以是更相关的管道的一部分。

先决条件

必须先完成以下步骤，然后才能使用 CLI：

1. 安装和配置命令行界面 (CLI)。有关更多信息，请参阅访问 AWS Data Pipeline。
2. 确保名为 DataPipelineDefaultRole 和 DataPipelineDefaultResourceRole 的 IAM 角色存在。AWS Data Pipeline 控制台会自动为您创建这些角色。如果您一次也没有使用过 AWS Data Pipeline 控制台，则必须手动创建这些角色。有关更多信息，请参阅适用于 AWS Data Pipeline 的 IAM 角色。

任务

- 上传并激活管道定义
- 监控管道运行

## 创建管道定义文件

以下代码是简单 Amazon EMR 集群的管道定义文件，该集群运行由 Amazon EMR 提供的现有 Hadoop 流式处理作业。该示例应用程序称作 WordCount，也可使用 Amazon EMR 控制台运行它。

将此代码复制到一个文本文件中并将其另存为 `MyEmrPipelineDefinition.json`。您应将 Amazon S3 存储桶位置替换为您拥有的 Amazon S3 存储桶的名称。您还应替换开始日期和结束日期。要立即启动集群，请将 `startDateTime` 设置为过去某天的日期，并将 `endDateTime` 设置为将来某天的日期。随后，AWS Data Pipeline 立即开始启动“过期”集群，以尝试解决它所认为的工作积压。此回填意味着，您无需等待一个小时即会看到 AWS Data Pipeline 启动其第一个集群。

```
{
  "objects": [
    {
      "id": "Hourly",
      "type": "Schedule",
      "startDateTime": "2012-11-19T07:48:00",
      "endDateTime": "2012-11-21T07:48:00",
      "period": "1 hours"
    },
    {
      "id": "MyCluster",
      "type": "EmrCluster",
      "masterInstanceType": "m1.small",
      "schedule": {
        "ref": "Hourly"
      }
    },
    {
      "id": "MyEmrActivity",
      "type": "EmrActivity",
      "schedule": {
        "ref": "Hourly"
      },
      "runsOn": {
        "ref": "MyCluster"
      },
```

```
      "step": "/home/hadoop/contrib/streaming/hadoop-streaming.jar,-input,s3n://
elasticmapreduce/samples/wordcount/input,-output,s3://myawsbucket/wordcount/
output/#{@scheduledStartTime},-mapper,s3n://elasticmapreduce/samples/wordcount/
wordSplitter.py,-reducer,aggregate"
    }
  ]
}
```

此管道有三个对象：

- `Hourly`，表示工作的计划。您可以将计划设置为活动上的字段之一。在执行此操作时，该活动将根据计划运行或 (在此示例中) 每小时运行一次。
- `MyCluster`，表示用于运行集群的 Amazon EC2 实例组。您可以指定要作为集群运行的 EC2 实例的大小和数目。如果您不指定实例数，则集群在启动时有两个节点：一个主节点和一个任务节点。您可以指定要在其中启动集群的子网。您可以向集群添加其他配置，例如，用于将其他软件加载到由 Amazon EMR 提供的 AMI 上的引导操作。
- `MyEmrActivity`，表示要使用集群处理的计算。Amazon EMR 支持多种类型的集群，包括流式处理、级联和脚本化 Hive。`runsOn` 字段重新引用 MyCluster，并将它用作集群基础的规范。

## 上传并激活管道定义

您必须上传您的管道定义并激活您的管道。在以下示例命令中，将 *`pipeline_name`* 替换为管道的标签，将 *`pipeline_file`* 替换为管道定义 `.json` 文件的完全限定路径。

AWS CLI

要创建管道定义并激活管道，请使用以下 create-pipeline 命令。记下您的管道 ID，因为您将在大多数 CLI 命令中使用这个值。

```
aws datapipeline create-pipeline --name pipeline_name --unique-id token
{
    "pipelineId": "df-00627471SOVYZEXAMPLE"
}
```

使用以下 put-pipeline-definition 命令更新管道定义。

```
aws datapipeline put-pipeline-definition --pipeline-id df-00627471SOVYZEXAMPLE --
pipeline-definition file://MyEmrPipelineDefinition.json
```

如果您的管道成功验证，则 `validationErrors` 字段为空。您应该查看所有警告。

要激活管道，请使用以下 activate-pipeline 命令。

```
aws datapipeline activate-pipeline --pipeline-id df-00627471SOVYZEXAMPLE
```

您可以使用以下 list-pipelines 命令来验证您的管道是否出现在管道列表中。

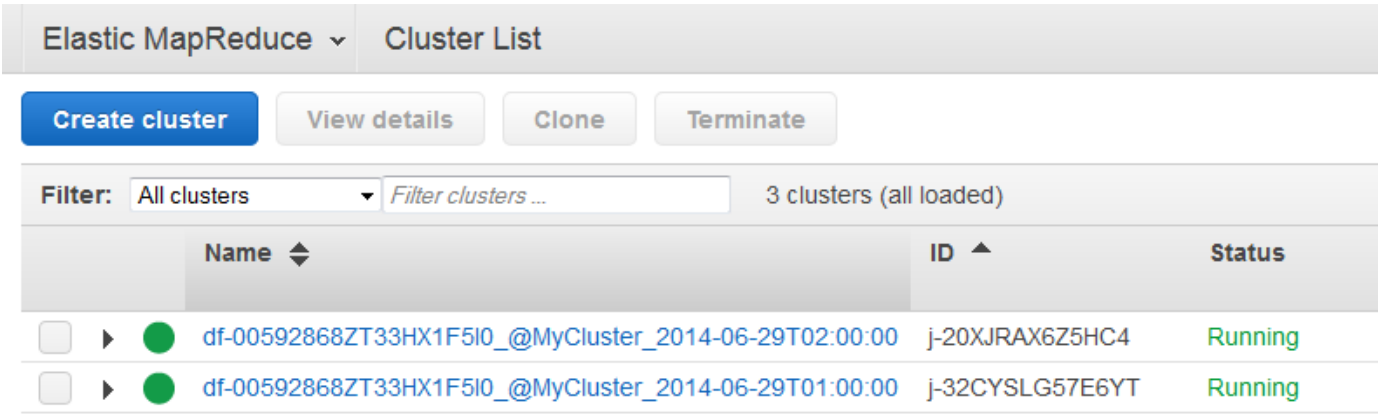
```
aws datapipeline list-pipelines
```

## 监控管道运行

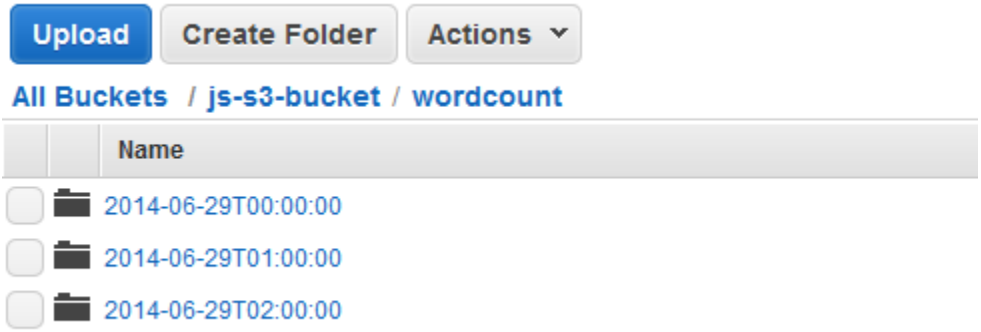
您可以使用 Amazon EMR 控制台查看由 AWS Data Pipeline 启动的集群，也可以使用 Amazon S3 控制台查看输出文件夹。

查看由 AWS Data Pipeline 启动的集群的进度

1. 打开 Amazon EMR 控制台。
2. 由 AWS Data Pipeline 生成的集群具有如下格式的名称：*<pipeline-identifier>*_@*<emr-cluster-name>*_*<launch-time>*。




3. 在某个运行完成后，请打开 Amazon S3 控制台并检查带时间戳的输出文件夹是否存在并且包含集群的预期结果。



# 使用 AWS Data Pipeline 在 Amazon S3 存储桶之间复制 CSV 数据

在您阅读了 什么是 AWS Data Pipeline？ 之后，并且确定希望使用 AWS Data Pipeline 自动执行数据的移动和转换时，即可开始创建数据管道。为了帮助您了解 AWS Data Pipeline 的工作原理，让我们演练一个简单的任务。

本教程将指导您完成创建数据管道的过程，以将数据从一个 Amazon S3 存储桶复制到另一个存储桶，然后在复制活动成功完成后发送 Amazon SNS 通知。您可以为此复制活动使用 AWS Data Pipeline 管理的 EC2 实例。

管道对象

管道使用以下对象：

CopyActivity

AWS Data Pipeline 为此管道执行的活动（将 CSV 数据从一个 Amazon S3 存储桶复制到另一个存储桶）。

> **Important**
>
> 通过 CopyActivity 和 S3DataNode 使用 CSV 文件格式时有限制。有关更多信息，请参阅CopyActivity。

计划

此活动的开始日期、时间和重复频率。您可以选择指定结束日期和时间。

Ec2Resource

AWS Data Pipeline 执行此活动使用的资源（EC2 实例）。

S3DataNode

此管道的输入和输出节点（Amazon S3 存储桶）。

SnsAlarm

满足指定条件时 AWS Data Pipeline 必须采取的操作（任务成功完成后将 Amazon SNS 通知发送到主题）。

目录



# 开始前的准备工作

确保您已完成以下步骤。

- 完成对 AWS Data Pipeline 进行设置中所述的任务。
- (可选) 为实例设置一个 VPC，并为 VPC 设置一个安全组。
- 创建 Amazon S3 存储桶作为数据源。

  有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的创建存储桶。
- 将数据上载到 Amazon S3 存储桶。

  有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的向存储桶添加对象。
- 创建另一个 Amazon S3 存储桶作为数据目标
- 创建用于发送电子邮件通知的主题并记下主题的 Amazon 资源名称 (ARN)。有关更多信息，请参阅《Amazon Simple Notification Service 入门指南》 中的创建主题。
- （可选）本教程使用 AWS Data Pipeline 创建的默认 IAM 角色策略。如果您更希望创建和配置您自己的 IAM 角色策略和信任关系，请按照 适用于 AWS Data Pipeline 的 IAM 角色 中所述的说明操作。

# 使用命令行复制 CSV 数据

您可以创建管道并使用管道将数据从一个 Amazon S3 存储桶复制到另一个存储桶。

先决条件

在开始本教程之前，您必须完成以下步骤：

1. 安装和配置命令行界面 (CLI)。有关更多信息，请参阅访问 AWS Data Pipeline。
2. 确保名为 DataPipelineDefaultRole 和 DataPipelineDefaultResourceRole 的 IAM 角色存在。AWS Data Pipeline 控制台会自动为您创建这些角色。如果您一次也没有使用过 AWS Data Pipeline 控制台，则必须手动创建这些角色。有关更多信息，请参阅适用于 AWS Data Pipeline 的 IAM 角色。

任务

- 以 JSON 格式定义管道
- 上传并激活管道定义

## 以 JSON 格式定义管道

此示例场景说明了如何使用 JSON 管道定义和 AWS Data Pipeline CLI，在两个 Amazon S3 存储桶之间计划按特定时间间隔复制数据。这是完整管道定义 JSON 文件，其每个部分后跟说明。

> **Note**
>
> 我们建议您使用文本编辑器，这可帮助您验证 JSON 格式文件的语法，并使用 .json 文件扩展名来命名文件。

在本示例中，为清晰起见，我们跳过可选字段，只显示必填字段。此示例的完整管道 JSON 文件为：

```
{
  "objects": [
    {
      "id": "MySchedule",
      "type": "Schedule",
      "startDateTime": "2013-08-18T00:00:00",
      "endDateTime": "2013-08-19T00:00:00",
      "period": "1 day"
    },
    {
      "id": "S3Input",
      "type": "S3DataNode",
      "schedule": {
        "ref": "MySchedule"
      },
      "filePath": "s3://example-bucket/source/inputfile.csv"
    },
    {
      "id": "S3Output",
      "type": "S3DataNode",
      "schedule": {
        "ref": "MySchedule"
      },
```

```
      "filePath": "s3://example-bucket/destination/outputfile.csv"
    },
    {
      "id": "MyEC2Resource",
      "type": "Ec2Resource",
      "schedule": {
        "ref": "MySchedule"
      },
      "instanceType": "m1.medium",
      "role": "DataPipelineDefaultRole",
      "resourceRole": "DataPipelineDefaultResourceRole"
    },
    {
      "id": "MyCopyActivity",
      "type": "CopyActivity",
      "runsOn": {
        "ref": "MyEC2Resource"
      },
      "input": {
        "ref": "S3Input"
      },
      "output": {
        "ref": "S3Output"
      },
      "schedule": {
        "ref": "MySchedule"
      }
    }
  ]
}
```

计划

管道定义一个计划，其中具有开始和结束日期，以及一个确定此管道中活动的运行频率的周期。

```
{
  "id": "MySchedule",
  "type": "Schedule",
  "startDateTime": "2013-08-18T00:00:00",
  "endDateTime": "2013-08-19T00:00:00",
  "period": "1 day"
},
```

Amazon S3 数据节点

然后，输入 S3DataNode 管道组件定义输入文件的位置；在本例中是 Amazon S3 存储桶位置。输入 S3DataNode 组件由以下字段定义：

```
{
  "id": "S3Input",
  "type": "S3DataNode",
  "schedule": {
    "ref": "MySchedule"
  },
  "filePath": "s3://example-bucket/source/inputfile.csv"
},
```

Id

输入位置的用户定义名称 (仅供您参考的标签)。

类型

管道组件类型，这是“S3DataNode”，与 Amazon S3 存储桶中数据所在位置匹配。

计划

对我们在 JSON 文件前面行中创建的标记为“MySchedule”的计划组件的引用。

路径

与数据节点关联的数据的路径。数据节点的语法取决于其类型。例如，Amazon S3 路径的语法遵循不适用于数据库表的其他语法。

接下来，输出 S3DataNode 组件定义数据的输出目标位置。它遵循与输入 S3DataNode 组件相同的格式，但组件名称不同，并使用不同路径指示目标文件。

```
{
  "id": "S3Output",
  "type": "S3DataNode",
  "schedule": {
    "ref": "MySchedule"
  },
  "filePath": "s3://example-bucket/destination/outputfile.csv"
},
```

资源

这是执行复制操作的计算资源的定义。在本示例中，AWS Data Pipeline 应该自动创建 EC2 实例来执行复制任务，并在任务完成后终止资源。此处定义的字段控制完成工作的 EC2 实例的创建操作及其功能。EC2Resource 由以下字段定义：

```
{
  "id": "MyEC2Resource",
  "type": "Ec2Resource",
  "schedule": {
    "ref": "MySchedule"
  },
  "instanceType": "m1.medium",
  "role": "DataPipelineDefaultRole",
  "resourceRole": "DataPipelineDefaultResourceRole"
},
```

Id

管道计划的用户定义名称，这是仅供您参考的标签。

类型

执行工作的计算资源的类型；在这种情况下为 EC2 实例。还有其他资源类型可用，如 EmrCluster 类型。

计划

根据它来创建此计算资源的计划。

instanceType

要创建的 EC2 实例的大小。请确保您设置了合适的 EC2 实例大小，能够最佳地符合您希望使用 AWS Data Pipeline 执行的工作负载量。在这种情况下，我们设置一个 m1.medium EC2 实例。有关不同实例类型以及何时使用各种实例的详细信息，请参阅位于 http://aws.amazon.com/ec2/instance-types/ 的 Amazon EC2 实例类型。

角色

访问资源的账户的 IAM 角色，例如访问 Amazon S3 存储桶检索数据。

resourceRole

创建资源的账户的 IAM 角色，如代表您创建和配置 EC2 实例。Role 和 ResourceRole 可以是相同角色，但在安全配置中可分别提供更细粒度。

活动

JSON 文件的最后一个部分是活动的定义，表示要执行的工作。此示例使用 CopyActivity 将数据从位于 http://aws.amazon.com/ec2/instance-types/ 存储桶中的 CSV 文件复制到另一个存储桶。CopyActivity 组件由以下字段定义：

```
{
  "id": "MyCopyActivity",
  "type": "CopyActivity",
  "runsOn": {
    "ref": "MyEC2Resource"
  },
  "input": {
    "ref": "S3Input"
  },
  "output": {
    "ref": "S3Output"
  },
  "schedule": {
    "ref": "MySchedule"
  }
}
```

Id

活动的用户定义名称，这是仅供您参考的标签。

类型

要执行的活动类型，例如 MyCopyActivity。

runsOn

执行此活动定义的工作的计算资源。在本示例中，我们提供了对之前定义的 EC2 实例的引用。使用 runsOn 字段会让 AWS Data Pipeline 为您创建 EC2 实例。runsOn 字段指示资源存在于 AWS 基础设施中，而 workerGroup 值指示您要使用自己的本地资源执行工作。

输入

待复制数据的位置。

输出

目标位置数据。

计划

运行此活动的计划。

### 上传并激活管道定义

您必须上传您的管道定义并激活您的管道。在以下示例命令中，将 *pipeline_name* 替换为管道的标签，将 *pipeline_file* 替换为管道定义 .json 文件的完全限定路径。

AWS CLI

要创建管道定义并激活管道，请使用以下 create-pipeline 命令。记下您的管道 ID，因为您将在大多数 CLI 命令中使用这个值。

```
aws datapipeline create-pipeline --name pipeline_name --unique-id token
{
    "pipelineId": "df-00627471SOVYZEXAMPLE"
}
```

使用以下 put-pipeline-definition 命令更新管道定义。

```
aws datapipeline put-pipeline-definition --pipeline-id df-00627471SOVYZEXAMPLE --
pipeline-definition file://MyEmrPipelineDefinition.json
```

如果您的管道成功验证，则 validationErrors 字段为空。您应该查看所有警告。

要激活管道，请使用以下 activate-pipeline 命令。

```
aws datapipeline activate-pipeline --pipeline-id df-00627471SOVYZEXAMPLE
```

您可以使用以下 list-pipelines 命令来验证您的管道是否出现在管道列表中。

```
aws datapipeline list-pipelines
```

## 使用 AWS Data Pipeline 将 MySQL 数据导出到 Amazon S3。

本教程指导您完成创建数据管道的过程，将数据（行）从 MySQL 数据库中的表复制到 Amazon S3 存储桶中的 CSV（逗号分隔值）文件，然后在复制活动成功完成后发送 Amazon SNS 通知。您将使用 AWS Data Pipeline 为此复制活动提供的 EC2 实例。

管道对象

管道使用以下对象：

- CopyActivity
- Ec2Resource
- MySqlDataNode
- S3DataNode
- SnsAlarm

目录



# 开始前的准备工作

确保您已完成以下步骤。

- 完成对 AWS Data Pipeline 进行设置中所述的任务。
- (可选) 为实例设置一个 VPC，并为 VPC 设置一个安全组。
- 创建 Amazon S3 存储桶作为数据输出。

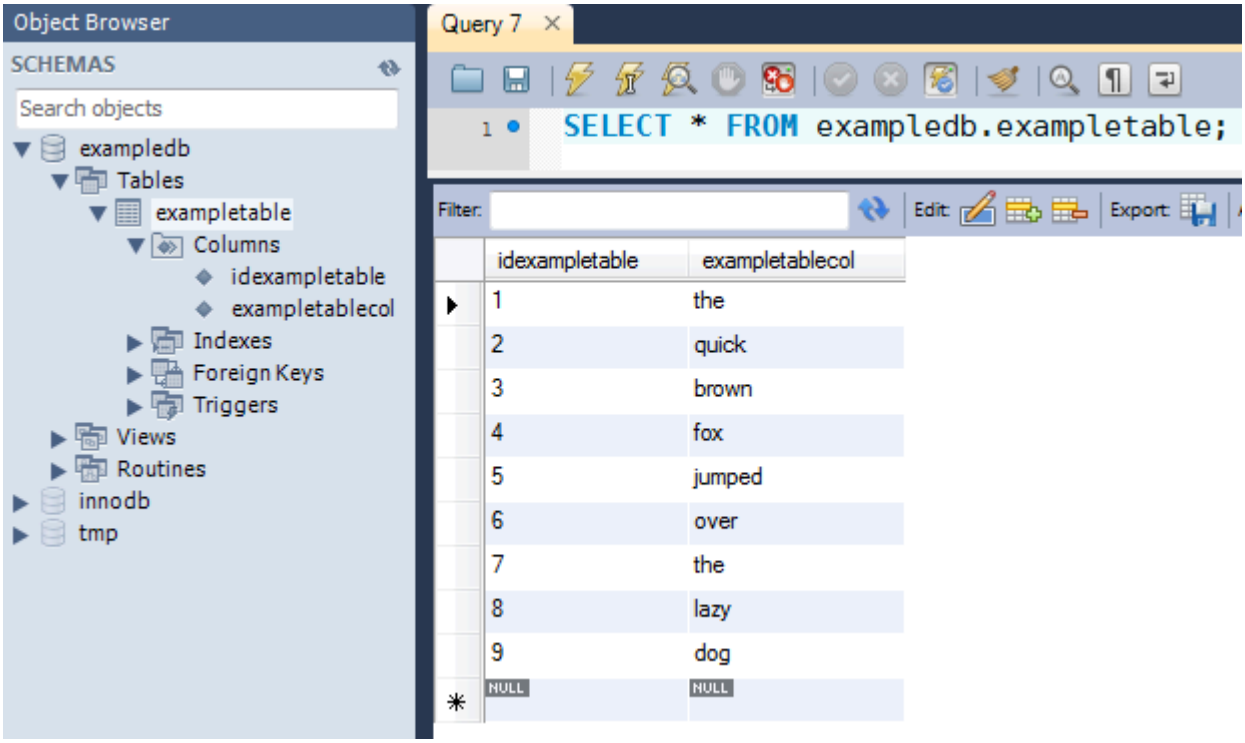
  有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的创建存储桶。
- 创建并启动 MySQL 数据库实例作为您的数据源。

  有关更多信息，请参阅Amazon RDS 入门指南中的 启动数据库实例。在您拥有了 Amazon RDS 实例之后，请参阅 MySQL 文档中的创建表。

  > **Note**
  >
  > 请记下创建 MySQL 实例使用的用户名和密码。在您启动 MySQL 数据库实例之后，请记录实例的终端节点。稍后您将需要此信息。

- 连接到 MySQL 数据库实例，创建表，然后将测试数据值添加到新创建的表中。

  为方便说明，我们使用具有以下配置和示例数据的 MySQL 表创建本教程。下面的屏幕截图来自 MySQL Workbench 5.2 CE：




有关更多信息，请参阅 MySQL 文档中的创建表和 MySQL Workbench 产品页面。

- 创建用于发送电子邮件通知的主题并记下主题的 Amazon 资源名称 (ARN)。有关更多信息，请参阅Amazon Simple Notification Service 入门指南中的创建主题。
- （可选）本教程使用 AWS Data Pipeline 创建的默认 IAM 角色策略。如果您更希望创建和配置自己的 IAM 角色策略和信任关系，请按照 适用于 AWS Data Pipeline 的 IAM 角色 中所述的说明操作。

## 使用命令行复制 MySQL 数据

您可以创建管道，将数据从 MySQL 表复制到 Amazon S3 存储桶中的文件。

先决条件

在开始本教程之前，您必须完成以下步骤：

1. 安装和配置命令行界面 (CLI)。有关更多信息，请参阅访问 AWS Data Pipeline。
2. 确保名为 DataPipelineDefaultRole 和 DataPipelineDefaultResourceRole 的 IAM 角色存在。AWS Data Pipeline 控制台会自动为您创建这些角色。如果您一次也没有使用过 AWS Data Pipeline 控制台，则必须手动创建这些角色。有关更多信息，请参阅适用于 AWS Data Pipeline 的 IAM 角色。
3. 设置 Amazon S3 存储桶和 Amazon RDS 实例。有关更多信息，请参阅开始前的准备工作。

任务

- 以 JSON 格式定义管道
- 上传并激活管道定义

## 以 JSON 格式定义管道

此示例场景显示如何使用 JSON 管道定义和 AWS Data Pipeline CLI 按指定的时间间隔，将数据（行）从 MySQL 数据库中的表复制到 Amazon S3 存储桶中的 CSV（逗号分隔值）文件。

这是完整管道定义 JSON 文件，其每个部分后跟说明。

> **Note**
>
> 我们建议您使用文本编辑器，这可帮助您验证 JSON 格式文件的语法，并使用 .json 文件扩展名来命名文件。

```
{
  "objects": [
    {
      "id": "ScheduleId113",
      "startDateTime": "2013-08-26T00:00:00",
      "name": "My Copy Schedule",
      "type": "Schedule",
      "period": "1 Days"
    },
    {
      "id": "CopyActivityId112",
      "input": {
        "ref": "MySqlDataNodeId115"
      },
      "schedule": {
        "ref": "ScheduleId113"
      },
      "name": "My Copy",
      "runsOn": {
        "ref": "Ec2ResourceId116"
      },
      "onSuccess": {
        "ref": "ActionId1"
      },
```

```
      "onFail": {
        "ref": "SnsAlarmId117"
      },
      "output": {
        "ref": "S3DataNodeId114"
      },
      "type": "CopyActivity"
    },
    {
      "id": "S3DataNodeId114",
      "schedule": {
        "ref": "ScheduleId113"
      },
      "filePath": "s3://example-bucket/rds-output/output.csv",
      "name": "My S3 Data",
      "type": "S3DataNode"
    },
    {
      "id": "MySqlDataNodeId115",
      "username": "my-username",
      "schedule": {
        "ref": "ScheduleId113"
      },
      "name": "My RDS Data",
      "*password": "my-password",
      "table": "table-name",
      "connectionString": "jdbc:mysql://your-sql-instance-name.id.region-
name.rds.amazonaws.com:3306/database-name",
      "selectQuery": "select * from #{table}",
      "type": "SqlDataNode"
    },
    {
      "id": "Ec2ResourceId116",
      "schedule": {
        "ref": "ScheduleId113"
      },
      "name": "My EC2 Resource",
      "role": "DataPipelineDefaultRole",
      "type": "Ec2Resource",
      "resourceRole": "DataPipelineDefaultResourceRole"
    },
    {
      "message": "This is a success message.",
      "id": "ActionId1",
```

```
      "subject": "RDS to S3 copy succeeded!",
      "name": "My Success Alarm",
      "role": "DataPipelineDefaultRole",
      "topicArn": "arn:aws:sns:us-east-1:123456789012:example-topic",
      "type": "SnsAlarm"
    },
    {
      "id": "Default",
      "scheduleType": "timeseries",
      "failureAndRerunMode": "CASCADE",
      "name": "Default",
      "role": "DataPipelineDefaultRole",
      "resourceRole": "DataPipelineDefaultResourceRole"
    },
    {
      "message": "There was a problem executing #{node.name} at for period
 #{node.@scheduledStartTime} to #{node.@scheduledEndTime}",
      "id": "SnsAlarmId117",
      "subject": "RDS to S3 copy failed",
      "name": "My Failure Alarm",
      "role": "DataPipelineDefaultRole",
      "topicArn": "arn:aws:sns:us-east-1:123456789012:example-topic",
      "type": "SnsAlarm"
    }
  ]
}
```

MySQL 数据节点

输入 MySqlDataNode 管道组件定义输入数据的位置；在本例中是 Amazon RDS 实例。输入 MySqlDataNode 组件由以下字段定义：

```
{
  "id": "MySqlDataNodeId115",
  "username": "my-username",
  "schedule": {
    "ref": "ScheduleId113"
  },
  "name": "My RDS Data",
  "*password": "my-password",
  "table": "table-name",
  "connectionString": "jdbc:mysql://your-sql-instance-name.id.region-
name.rds.amazonaws.com:3306/database-name",
```

```
  "selectQuery": "select * from #{table}",
  "type": "SqlDataNode"
},
```

Id

用户定义名称，这是仅供您参考的标签。

Username

数据库账户的用户名，该账户具有足够的权限从数据库表检索数据。使用您用户的名称替换 *my-username*。

计划

对我们在 JSON 文件前面行中创建的计划组件的引用。

名称

用户定义名称，这是仅供您参考的标签。

*密码

数据库账户密码带有星号前缀，指示 AWS Data Pipeline 必须加密密码值。使用您的用户的正确密码替换 *my-password*。密码字段前面带有星号特殊字符。有关更多信息，请参阅特殊字符。

表

包含要复制的数据的数据库表的名称。使用您的数据库表名称替换 *table-name*。

connectionString

连接到数据库的 CopyActivity 对象的 JDBC 连接字符串。

selectQuery

有效的 SQL SELECT 查询，用于指定要从数据库表复制的数据。请注意，#{table} 是重新使用在 JSON 文件前面行中“table”变量提供的表名的表达式。

类型

SqlDataNode 类型，在本例中是使用 MySQL 的 Amazon RDS 实例。

> **Note**
>
> MySqlDataNode 类型已被弃用。尽管您仍然可以使用 MySqlDataNode，我们建议您使用 SqlDataNode。

Amazon S3 数据节点

接下来，S3Output 管道组件定义输出文件的位置；在这种情况下是 Amazon S3 存储桶位置中的 CSV 文件。输出 S3DataNode 组件由以下字段定义：

```
{
  "id": "S3DataNodeId114",
  "schedule": {
    "ref": "ScheduleId113"
  },
  "filePath": "s3://example-bucket/rds-output/output.csv",
  "name": "My S3 Data",
  "type": "S3DataNode"
},
```

Id

用户定义 ID，这是仅供您参考的标签。

计划

对我们在 JSON 文件前面行中创建的计划组件的引用。

filePath

与数据节点关联的数据的路径，在本示例中是一个 CSV 输出文件。

名称

用户定义名称，这是仅供您参考的标签。

类型

管道对象类型，为 S3DataNode，与 Amazon S3 存储桶中数据所在位置匹配。

资源

这是执行复制操作的计算资源的定义。在本示例中，AWS Data Pipeline 应该自动创建 EC2 实例来执行复制任务，并在任务完成后终止资源。此处定义的字段控制完成工作的 EC2 实例的创建操作及其功能。EC2Resource 由以下字段定义：

```
{
  "id": "Ec2ResourceId116",
```

```
  "schedule": {
    "ref": "ScheduleId113"
  },
  "name": "My EC2 Resource",
  "role": "DataPipelineDefaultRole",
  "type": "Ec2Resource",
  "resourceRole": "DataPipelineDefaultResourceRole"
},
```

Id

用户定义 ID，这是仅供您参考的标签。

计划

根据它来创建此计算资源的计划。

名称

用户定义名称，这是仅供您参考的标签。

角色

访问资源的账户的 IAM 角色，例如访问 Amazon S3 存储桶检索数据。

类型

执行工作的计算资源的类型；在这种情况下为 EC2 实例。还有其他资源类型可用，如 EmrCluster 类型。

resourceRole

创建资源的账户的 IAM 角色，如代表您创建和配置 EC2 实例。Role 和 ResourceRole 可以是相同角色，但在安全配置中可分别提供更细粒度。

活动

JSON 文件的最后一个部分是活动的定义，表示要执行的工作。在本例中，我们使用 CopyActivity 组件将数据从 Amazon S3 存储桶中的文件复制到另一个文件。CopyActivity 组件由以下字段定义：

```
{
  "id": "CopyActivityId112",
  "input": {
    "ref": "MySqlDataNodeId115"
```

```
  },
  "schedule": {
    "ref": "ScheduleId113"
  },
  "name": "My Copy",
  "runsOn": {
    "ref": "Ec2ResourceId116"
  },
  "onSuccess": {
    "ref": "ActionId1"
  },
  "onFail": {
    "ref": "SnsAlarmId117"
  },
  "output": {
    "ref": "S3DataNodeId114"
  },
  "type": "CopyActivity"
},
```

Id

用户定义 ID，这是仅供您参考的标签

输入

待复制 MySQL 数据的位置

计划

运行此活动的计划

名称

用户定义名称，这是仅供您参考的标签

runsOn

执行此活动定义的工作的计算资源。在本示例中，我们提供了对之前定义的 EC2 实例的引用。使用 runs0n 字段会让 AWS Data Pipeline 为您创建 EC2 实例。runs0n 字段指示资源存在于 AWS 基础设施中，而 workerGroup 值指示您要使用自己的本地资源执行工作。

onSuccess

活动成功完成时发送的 SnsAlarm

onFail

活动失败时发送的 SnsAlarm

输出

CSV 输出文件的 Amazon S3 位置

类型

要执行的活动类型。

## 上传并激活管道定义

您必须上传您的管道定义并激活您的管道。在以下示例命令中，将 *pipeline_name* 替换为管道的标签，将 *pipeline_file* 替换为管道定义 .json 文件的完全限定路径。

AWS CLI

要创建管道定义并激活管道，请使用以下 create-pipeline 命令。记下您的管道 ID，因为您将在大多数 CLI 命令中使用这个值。

```
aws datapipeline create-pipeline --name pipeline_name --unique-id token
{
    "pipelineId": "df-00627471SOVYZEXAMPLE"
}
```

使用以下 put-pipeline-definition 命令更新管道定义。

```
aws datapipeline put-pipeline-definition --pipeline-id df-00627471SOVYZEXAMPLE --
pipeline-definition file://MyEmrPipelineDefinition.json
```

如果您的管道成功验证，则 validationErrors 字段为空。您应该查看所有警告。

要激活管道，请使用以下 activate-pipeline 命令。

```
aws datapipeline activate-pipeline --pipeline-id df-00627471SOVYZEXAMPLE
```

您可以使用以下 list-pipelines 命令来验证您的管道是否出现在管道列表中。

```
aws datapipeline list-pipelines
```

# 使用 AWS Data Pipeline 复制数据到 Amazon Redshift

本教程将指导您使用 AWS Data Pipeline 控制台中的复制到 Redshift 模板或者通过 AWS Data Pipeline CLI 使用管道定义文件完成创建管道的过程，该管道定期将数据从 Amazon S3 移至 Amazon Redshift。

Amazon S3 是一项 Web 服务，可让您在云中存储数据。有关更多信息，请参阅 Amazon Simple Storage Service 用户指南。

Amazon Redshift 是云中的数据仓库服务。有关更多信息，请参阅 Amazon Redshift 管理指南。

本教程有几个先决条件。完成以下步骤后，您可以使用控制台或 CLI 继续本教程。

目录



## 开始之前：配置 COPY 选项并加载数据

在 AWS Data Pipeline 中将数据复制到 Amazon Redshift 之前，请确保您已：

- 从 Amazon S3 加载数据。
- 在 Amazon Redshift 中设置 COPY 活动。

一旦您让这些选项生效并成功完成数据加载后，将这些选项传输到 AWS Data Pipeline 以便在其中执行复制操作。

有关 COPY 选项，请参阅 Amazon Redshift 数据库开发人员指南中的 COPY。

有关从 Amazon S3 加载数据的步骤，请参阅 Amazon Redshift 数据库开发人员指南中的从 Amazon S3 加载数据。

例如，Amazon Redshift 中的以下 SQL 命令会创建一个名为 LISTING 的新表，并从 Amazon S3 的公开可用的存储桶中复制示例数据。

将 <iam-role-arn> 和区域替换为您自己的值。

有关此示例的详细信息，请参阅 Amazon Redshift 入门指南中的从 Amazon S3 中加载示例数据。

```
create table listing(
 listid integer not null distkey,
 sellerid integer not null,
 eventid integer not null,
 dateid smallint not null  sortkey,
 numtickets smallint not null,
 priceperticket decimal(8,2),
 totalprice decimal(8,2),
 listtime timestamp);

copy listing from 's3://awssampledbuswest2/tickit/listings_pipe.txt'
credentials 'aws_iam_role=<iam-role-arn>'
delimiter '|' region 'us-west-2';
```

## 设置管道，创建安全组，并创建 Amazon Redshift 集群

为教程进行设置

1. 完成对 AWS Data Pipeline 进行设置中所述的任务。
2. 创建安全组。
   a. 打开 Amazon EC2 控制台。
   b. 在导航窗格中，单击 Security Groups (安全组)。
   c. 单击 Create Security Group。
   d. 为安全组指定名称和描述。
   e. [EC2-Classic] 为 No  VPCVPC 选择 。
   f. [EC2-VPC] 为 VPC 选择您 VPC 的 ID。
   g. 单击 Create。
3. [EC2-Classic] 创建 Amazon Redshift 集群安全组，并指定 Amazon EC2 安全组。
   a. 打开 Amazon Redshift 控制台。
   b. 在导航窗格中，单击 Security Groups (安全组)。
   c. 单击 Create Cluster Security Group。
   d. 在 Create Cluster Security Group 对话框中，为集群安全组指定名称和描述。
   e. 单击新集群安全组的名称。

f. 单击 Add Connection Type。

g. 在 Add Connection Type 对话框中，从 Connection Type 选择 EC2 Security Group，选择您从 EC2 Security Group Name 创建的安全组，然后单击 Authorize。

4. [EC2-VPC] 创建 Amazon Redshift 集群安全组，并指定 VPC 安全组。

a. 打开 Amazon EC2 控制台。

b. 在导航窗格中，单击 Security Groups (安全组)。

c. 单击 Create Security Group。

d. 在 Create Security Group 对话框中，指定安全组的名称和描述，然后为 VPC 选择您 VPC 的 ID。

e. 单击“Add Rule”。指定类型、协议和端口范围，然后开始在 Source 中键入安全组的 ID。选择您在第二步中创建的安全组。

f. 单击 Create。

5. 下面概括介绍了步骤。

如果您有一个现有的 Amazon Redshift 集群，请记下集群 ID。

要创建新集群和加载示例数据，请按照 Amazon Redshift 入门中的步骤操作。有关创建集群的更多信息，请参阅 Amazon Redshift 管理指南中的创建集群。

a. 打开 Amazon Redshift 控制台。

b. 单击 Launch Cluster。

c. 为您的集群提供所需的详细信息，然后单击 Continue (继续)。

d. 提供节点配置，然后单击 Continue。

e. 在其他配置信息的页面上，选择您创建的集群安全组，然后单击 Continue。

f. 检查您的集群规格，然后单击 Launch Cluster (启动集群)。

## 使用命令行复制数据到 Amazon Redshift

本教程演示如何将数据从 Amazon S3 复制到 Amazon Redshift。您将在 Amazon Redshift 中创建一个新表，然后使用 AWS Data Pipeline 将数据从公有 Amazon S3 存储桶传输到此表，该存储桶中包含 CSV 格式的示例输入数据。日志保存到您拥有的 Amazon S3 存储桶。

Amazon S3 是一项 Web 服务，可让您在云中存储数据。有关更多信息，请参阅 Amazon Simple Storage Service 用户指南。Amazon Redshift 是云中的数据仓库服务。有关更多信息，请参阅 Amazon Redshift 管理指南。

先决条件

在开始本教程之前，您必须完成以下步骤：

1. 安装和配置命令行界面 (CLI)。有关更多信息，请参阅访问 AWS Data Pipeline。
2. 确保名为 DataPipelineDefaultRole 和 DataPipelineDefaultResourceRole 的 IAM 角色存在。AWS Data Pipeline 控制台会自动为您创建这些角色。如果您一次也没有使用过 AWS Data Pipeline 控制台，则必须手动创建这些角色。有关更多信息，请参阅适用于 AWS Data Pipeline 的 IAM 角色。
3. 在 Amazon Redshift 中设置 COPY 命令，因为当您在 AWS Data Pipeline 中执行复制操作时，需要让这些相同的选项生效。有关信息，请参阅 开始之前：配置 COPY 选项并加载数据。
4. 设置 Amazon Redshift 数据库。有关更多信息，请参阅设置管道，创建安全组，并创建 Amazon Redshift 集群。

任务

- 以 JSON 格式定义管道
- 上传并激活管道定义

## 以 JSON 格式定义管道

此示例情景说明如何将数据从 Amazon S3 存储桶复制到 Amazon Redshift。

这是完整管道定义 JSON 文件，其每个部分后跟说明。我们建议您使用文本编辑器，这可帮助您验证 JSON 格式文件的语法，并使用 `.json` 文件扩展名来命名文件。

```
{
  "objects": [
    {
      "id": "CSVId1",
      "name": "DefaultCSV1",
      "type": "CSV"
    },
    {
      "id": "RedshiftDatabaseId1",
```

```
      "databaseName": "dbname",
      "username": "user",
      "name": "DefaultRedshiftDatabase1",
      "*password": "password",
      "type": "RedshiftDatabase",
      "clusterId": "redshiftclusterId"
    },
    {
      "id": "Default",
      "scheduleType": "timeseries",
      "failureAndRerunMode": "CASCADE",
      "name": "Default",
      "role": "DataPipelineDefaultRole",
      "resourceRole": "DataPipelineDefaultResourceRole"
    },
    {
      "id": "RedshiftDataNodeId1",
      "schedule": {
        "ref": "ScheduleId1"
      },
      "tableName": "orders",
      "name": "DefaultRedshiftDataNode1",
      "createTableSql": "create table StructuredLogs (requestBeginTime CHAR(30)
PRIMARY KEY DISTKEY SORTKEY, requestEndTime CHAR(30), hostname CHAR(100), requestDate
varchar(20));",
      "type": "RedshiftDataNode",
      "database": {
        "ref": "RedshiftDatabaseId1"
      }
    },
    {
      "id": "Ec2ResourceId1",
      "schedule": {
        "ref": "ScheduleId1"
      },
      "securityGroups": "MySecurityGroup",
      "name": "DefaultEc2Resource1",
      "role": "DataPipelineDefaultRole",
      "logUri": "s3://myLogs",
      "resourceRole": "DataPipelineDefaultResourceRole",
      "type": "Ec2Resource"
    },
    {
      "id": "ScheduleId1",
```

```
      "startDateTime": "yyyy-mm-ddT00:00:00",
      "name": "DefaultSchedule1",
      "type": "Schedule",
      "period": "period",
      "endDateTime": "yyyy-mm-ddT00:00:00"
    },
    {
      "id": "S3DataNodeId1",
      "schedule": {
        "ref": "ScheduleId1"
      },
      "filePath": "s3://datapipeline-us-east-1/samples/hive-ads-samples.csv",
      "name": "DefaultS3DataNode1",
      "dataFormat": {
        "ref": "CSVId1"
      },
      "type": "S3DataNode"
    },
    {
      "id": "RedshiftCopyActivityId1",
      "input": {
        "ref": "S3DataNodeId1"
      },
      "schedule": {
        "ref": "ScheduleId1"
      },
      "insertMode": "KEEP_EXISTING",
      "name": "DefaultRedshiftCopyActivity1",
      "runsOn": {
        "ref": "Ec2ResourceId1"
      },
      "type": "RedshiftCopyActivity",
      "output": {
        "ref": "RedshiftDataNodeId1"
      }
    }
  ]
}
```

有关这些对象的更多信息，请参阅以下文档。

对象

- 数据节点

- 资源
- 活动

数据节点

此示例使用输入数据节点、输出数据节点和数据库。

输入数据节点

输入 S3DataNode 管道组件定义了 Amazon S3 中的输入数据位置和输入数据的数据格式。有关更多信息，请参阅S3DataNode。

此输入组件由以下字段定义：

```
{
  "id": "S3DataNodeId1",
  "schedule": {
    "ref": "ScheduleId1"
  },
  "filePath": "s3://datapipeline-us-east-1/samples/hive-ads-samples.csv",
  "name": "DefaultS3DataNode1",
  "dataFormat": {
    "ref": "CSVId1"
  },
  "type": "S3DataNode"
},
```

`id`

用户定义 ID，这是仅供您参考的标签。

`schedule`

对计划组件的引用。

`filePath`

与数据节点关联的数据的路径，在本示例中是一个 CSV 输入文件。

`name`

用户定义名称，这是仅供您参考的标签。

dataFormat

对活动要处理的数据格式的引用。

输出数据节点

输出 RedshiftDataNode 管道组件定义了输出数据的位置；在本例中是 Amazon Redshift 数据库中的表。有关更多信息，请参阅RedshiftDataNode。此输出组件由以下字段定义：

```
{
  "id": "RedshiftDataNodeId1",
  "schedule": {
    "ref": "ScheduleId1"
  },
  "tableName": "orders",
  "name": "DefaultRedshiftDataNode1",
  "createTableSql": "create table StructuredLogs (requestBeginTime CHAR(30) PRIMARY
 KEY DISTKEY SORTKEY, requestEndTime CHAR(30), hostname CHAR(100), requestDate
 varchar(20));",
  "type": "RedshiftDataNode",
  "database": {
    "ref": "RedshiftDatabaseId1"
  }
},
```

id

用户定义 ID，这是仅供您参考的标签。

schedule

对计划组件的引用。

tableName

Amazon Redshift 表的名称。

name

用户定义名称，这是仅供您参考的标签。

createTableSql

一个用于在数据库中创建表的 SQL 表达式。

database

对 Amazon Redshift 数据库的引用。

数据库

RedshiftDatabase 组件由以下字段定义。有关更多信息，请参阅RedshiftDatabase。

```
{
  "id": "RedshiftDatabaseId1",
  "databaseName": "dbname",
  "username": "user",
  "name": "DefaultRedshiftDatabase1",
  "*password": "password",
  "type": "RedshiftDatabase",
  "clusterId": "redshiftclusterId"
},
```

id

用户定义 ID，这是仅供您参考的标签。

databaseName

逻辑数据库的名称。

username

连接到数据库的用户名。

name

用户定义名称，这是仅供您参考的标签。

password

连接到数据库的密码。

clusterId

Redshift 集群的 ID。

资源

这是执行复制操作的计算资源的定义。在本示例中，AWS Data Pipeline 应该自动创建 EC2 实例来执行复制任务，并在任务完成后终止实例。此处定义的字段控制完成任务的实例的创建和功能。有关更多信息，请参阅Ec2Resource。

Ec2Resource 由以下字段定义：

```
{
  "id": "Ec2ResourceId1",
  "schedule": {
    "ref": "ScheduleId1"
  },
  "securityGroups": "MySecurityGroup",
  "name": "DefaultEc2Resource1",
  "role": "DataPipelineDefaultRole",
  "logUri": "s3://myLogs",
  "resourceRole": "DataPipelineDefaultResourceRole",
  "type": "Ec2Resource"
},
```

id

用户定义 ID，这是仅供您参考的标签。

schedule

根据它来创建此计算资源的计划。

securityGroups

要用于资源池中实例的安全组。

name

用户定义名称，这是仅供您参考的标签。

role

访问资源的账户的 IAM 角色，例如访问 Amazon S3 存储桶检索数据。

logUri

从 Ec2Resource 备份任务运行程序日志的 Amazon S3 目标路径。

`resourceRole`

创建资源的账户的 IAM 角色，如代表您创建和配置 EC2 实例。Role 和 ResourceRole 可以是相同角色，但在安全配置中可分别提供更细粒度。

活动

JSON 文件的最后一个部分是活动的定义，表示要执行的工作。在本例中，我们使用 `RedshiftCopyActivity` 组件将数据从 Amazon S3 复制到 Amazon Redshift。有关更多信息，请参阅RedshiftCopyActivity。

`RedshiftCopyActivity` 组件由以下字段定义：

```
{
  "id": "RedshiftCopyActivityId1",
  "input": {
    "ref": "S3DataNodeId1"
  },
  "schedule": {
    "ref": "ScheduleId1"
  },
  "insertMode": "KEEP_EXISTING",
  "name": "DefaultRedshiftCopyActivity1",
  "runsOn": {
    "ref": "Ec2ResourceId1"
  },
  "type": "RedshiftCopyActivity",
  "output": {
    "ref": "RedshiftDataNodeId1"
  }
},
```

`id`

用户定义 ID，这是仅供您参考的标签。

`input`

对 Amazon S3 源文件的引用。

`schedule`

运行此活动的计划。

`insertMode`

插入类型 (KEEP_EXISTING、OVERWRITE_EXISTING 或 TRUNCATE)。

`name`

用户定义名称，这是仅供您参考的标签。

`runsOn`

执行此活动定义的工作的计算资源。

`output`

对 Amazon Redshift 目标表的引用。

## 上传并激活管道定义

您必须上传您的管道定义并激活您的管道。在以下示例命令中，将 *pipeline_name* 替换为管道的标签，将 *pipeline_file* 替换为管道定义 `.json` 文件的完全限定路径。

AWS CLI

要创建管道定义并激活管道，请使用以下 create-pipeline 命令。记下您的管道 ID，因为您将在大多数 CLI 命令中使用这个值。

```
aws datapipeline create-pipeline --name pipeline_name --unique-id token
{
    "pipelineId": "df-00627471SOVYZEXAMPLE"
}
```

使用以下 put-pipeline-definition 命令更新管道定义。

```
aws datapipeline put-pipeline-definition --pipeline-id df-00627471SOVYZEXAMPLE --
pipeline-definition file://MyEmrPipelineDefinition.json
```

如果您的管道成功验证，则 `validationErrors` 字段为空。您应该查看所有警告。

要激活管道，请使用以下 activate-pipeline 命令。

```
aws datapipeline activate-pipeline --pipeline-id df-00627471SOVYZEXAMPLE
```

您可以使用以下 list-pipelines 命令来验证您的管道是否出现在管道列表中。

```
aws datapipeline list-pipelines
```

# 管道表达式和函数

本部分介绍在管道中使用表达式和函数的语法，包括相关的数据类型。

## 简单数据类型

以下类型的数据可设置为字段值。

类型

- DateTime
- 数值
- 对象引用
- 周期
- 字符串

### DateTime

AWS Data Pipeline 仅支持使用 UTC/GMT 的“YYYY-MM-DDTHH:MM:SS”格式表示的日期和时间。以下示例将 Schedule 对象的 `startDateTime` 字段设置为 `1/15/2012, 11:59 p.m.` (在 UTC/GMT 时区内)。

```
"startDateTime" : "2012-01-15T23:59:00"
```

### 数值

AWS Data Pipeline 支持整数和浮点值。

### 对象引用

管道定义中的对象。这可以是当前对象 (管道中的其他位置定义的对象名称) 或一个在字段中列出当前对象的对象 (由 node 关键字引用)。有关 node 的更多信息，请参阅 引用字段和对象。有关管道对象类型的更多信息，请参阅管道对象引用。

### 周期

指示已计划事件的运行频率。它用格式“N [`years|months|weeks|days|hours|minutes`]”表示，其中 N 是正整数值。

最短时段为 15 分钟，最长时段为 3 年。

以下示例将 Schedule 对象的 period 字段设置为 3 小时。这会创建一个每 3 小时运行一次的计划。

```
"period" : "3 hours"
```

## 字符串

标准字符串值。字符串必须用双引号 (") 引起来。您可以使用反斜杠字符 (\) 对字符串中的字符进行转义。不支持多行字符串。

以下示例显示了 id 字段的有效字符串值的示例。

```
"id" : "My Data Object"

"id" : "My \"Data\" Object"
```

字符串还可以包含计算结果为字符串值的表达式。它们将插入字符串中，并用“#{”和“}”分隔开。以下示例使用表达式将当前对象的名称插入路径中。

```
"filePath" : "s3://myBucket/#{name}.csv"
```

有关使用表达式的更多信息，请参阅引用字段和对象和表达式计算。

# 表达式

利用表达式，您可以跨相关对象共享值。表达式由 AWS Data Pipeline Web 服务在运行时处理，确保所有表达式都替代为表达式值。

表达式用“#{”和“}”分隔。您可以在字符串合法的任何管道定义对象中使用表达式。如果槽是一个引用，或者其类型为 ID、NAME、TYPE 或 SPHERE 之一，则不计算其值并且将按字面使用它。

以下表达式调用 AWS Data Pipeline 函数之一。有关更多信息，请参阅表达式计算。

```
#{format(myDateTime,'YYYY-MM-dd hh:mm:ss')}
```

## 引用字段和对象

表达式既可以使用其所在的当前对象的字段，也可以使用通过引用链接的另一个对象的字段。

槽格式由创建时间后跟对象创建时间组成，例如 @S3BackupLocation_2018-01-31T11:05:33。

您也可以引用管道定义中指定的确切的槽 ID，例如 Amazon S3 备份位置的槽 ID。要引用槽 ID，请使用 #{parent.@id}。

在以下示例中，filePath 字段引用同一对象中的 id 字段来生成文件名。filePath 值的计算结果为“s3://mybucket/ExampleDataNode.csv”。

```
{
  "id" : "ExampleDataNode",
  "type" : "S3DataNode",
  "schedule" : {"ref" : "ExampleSchedule"},
  "filePath" : "s3://mybucket/#{parent.@id}.csv",
  "precondition" : {"ref" : "ExampleCondition"},
  "onFail" : {"ref" : "FailureNotify"}
}
```

要使用通过引用链接的另一个对象上存在的字段，请使用 node 关键字。此关键字仅适用于警报和先决条件对象。

继续前一个示例，SnsAlarm 中的表达式可引用 Schedule 中的日期和时间范围，因为 S3DataNode 将引用二者。

具体而言，FailureNotify 的 message 字段可从 ExampleSchedule 中使用 @scheduledStartTime 和 @scheduledEndTime 运行时字段，因为 ExampleDataNode 的 onFail 字段引用 FailureNotify，其 schedule 字段引用 ExampleSchedule。

```
{
    "id" : "FailureNotify",
    "type" : "SnsAlarm",
    "subject" : "Failed to run pipeline component",
    "message": "Error for interval
 #{node.@scheduledStartTime}..#{node.@scheduledEndTime}.",
    "topicArn":"arn:aws:sns:us-east-1:28619EXAMPLE:ExampleTopic"
},
```

> **Note**
>
> 您可以创建具有依赖项的管道，例如，管道中的任务，这些任务依赖于其他系统或任务的工作。如果管道需要某些资源，请使用与数据节点和任务关联的先决条件将这些依赖项添加到管

道。这可使管道更易于调试且更具弹性。此外，由于难以跨管道排查问题，因此，请将依赖项保留在单个管道中 (如果可能)。

## 嵌套表达式

AWS Data Pipeline 使您能够嵌入值来创建更复杂的表达式。例如，要执行时间计算 (从 scheduledStartTime 中减去 30 分钟) 并设置结果的格式以在管道定义中使用，您可以在活动中使用以下表达式：

```
#{format(minusMinutes(@scheduledStartTime,30),'YYYY-MM-dd hh:mm:ss')}
```

并使用 node 前缀 (如果表达式是 SnsAlarm 或 Precondition 的一部分)：

```
#{format(minusMinutes(node.@scheduledStartTime,30),'YYYY-MM-dd hh:mm:ss')}
```

## 列表

可基于列表和列表中的函数计算表达式。例如，假定按如下所示定义列表："myList": ["one","two"]。如果在表达式 #{'this is ' + myList} 中使用此列表，则该表达式的计算结果将为 ["this is one", "this is two"]。如果您有两个列表，Data Pipeline 最终将在计算中平展列表。例如，如果 myList1 定义为 [1,2] 且 myList2 定义为 [3,4]，则表达式 [#{myList1}, #{myList2}] 的计算结果将为 [1,2,3,4]。

## 节点表达式

对于对管道组件的父对象的向后引用，AWS Data Pipeline 在 SnsAlarm 或 PreCondition 中使用 #{node.*} 表达式。由于 SnsAlarm 和 PreCondition 引用自一个活动或资源且未从其向后引用，因此，node 提供了对引用站点的引用方式。例如，以下管道定义演示故障通知如何使用 node 来引用其父级 (在此示例中，为 ShellCommandActivity)，并在 SnsAlarm 消息中包含父级的计划的开始和结束时间。ShellCommandActivity 上的 scheduledStartTime 引用不需要 node 前缀，因为 scheduledStartTime 将引用自身。

> **Note**
>
> 字段前面的 @ 符号指示这些字段是运行时字段。

```
{
  "id" : "ShellOut",
  "type" : "ShellCommandActivity",
  "input" : {"ref" : "HourlyData"},
  "command" : "/home/userName/xxx.sh #{@scheduledStartTime} #{@scheduledEndTime}",
  "schedule" : {"ref" : "HourlyPeriod"},
  "stderr" : "/tmp/stderr:#{@scheduledStartTime}",
  "stdout" : "/tmp/stdout:#{@scheduledStartTime}",
  "onFail" : {"ref" : "FailureNotify"},
},
{
  "id" : "FailureNotify",
  "type" : "SnsAlarm",
  "subject" : "Failed to run pipeline component",
  "message": "Error for interval
 #{node.@scheduledStartTime}..#{node.@scheduledEndTime}.",
  "topicArn":"arn:aws:sns:us-east-1:28619EXAMPLE:ExampleTopic"
},
```

AWS Data Pipeline 支持用户定义的字段而非运行时字段的可传递引用。可传递引用是两个管道组件之间的引用，该引用依靠另一个管道组件作为中介。以下示例显示了一个对可传递的用户定义的字段的引用和一个对不可传递的运行时字段的引用，二者均有效。有关更多信息，请参阅用户定义字段。

```
{
  "name": "DefaultActivity1",
  "type": "CopyActivity",
  "schedule": {"ref": "Once"},
  "input": {"ref": "s3nodeOne"},
  "onSuccess": {"ref": "action"},
  "workerGroup": "test",
  "output": {"ref": "s3nodeTwo"}
},
{
  "name": "action",
  "type": "SnsAlarm",
  "message": "S3 bucket '#{node.output.directoryPath}' succeeded at
 #{node.@actualEndTime}.",
  "subject": "Testing",
  "topicArn": "arn:aws:sns:us-east-1:28619EXAMPLE:ExampleTopic",
  "role": "DataPipelineDefaultRole"
}
```

## 表达式计算

AWS Data Pipeline 提供了一组函数，可用于计算字段的值。以下示例使用 makeDate 函数将 Schedule 对象的 startDateTime 字段设置为 "2011-05-24T0:00:00" GMT/UTC。

```
"startDateTime" : "makeDate(2011,5,24)"
```

## 数学函数

以下函数可用于使用数字值。

| 函数 | 描述 |
| --- | --- |
| + | 加。<br>示例：#{1 + 2}<br>结果：3 |
| - | 减。<br>示例：#{1 - 2}<br>结果：-1 |
| * | 乘。<br>示例：#{1 * 2}<br>结果：2 |
| / | 除。如果您将两个整数相除，结果将被截断。<br>示例：#{1 / 2}，结果：0<br>示例：#{1.0 / 2}，结果：.5 |
| ^ | 指数。<br>示例：#{2 ^ 2} |

| 函数 | 描述 |
| --- | --- |
| | 结果：4.0 |

## 字符串函数

以下函数可用于使用字符串值。

| 函数 | 描述 |
| --- | --- |
| + | 联接。非字符串值会先转换为字符串。<br>示例：#{"hel" + "lo"}<br>结果："hello" |

## 日期和时间函数

以下函数可用于使用 DateTime 值。在示例中，myDateTime 的值为 May 24, 2011 @ 5:10 pm GMT。

> **Note**
>
> AWS Data Pipeline 的日期/时间格式为 Joda Time，后者用于替代 Java 日期和时间类。有关更多信息，请参阅 Joda Time - 类 DateTimeFormat。

| 函数 | 描述 |
| --- | --- |
| int day(DateTime myDateTime) | 获取 DateTime 值的天 (用整数表示)。<br>示例：#{day(myDateTime)}<br>结果：24 |

| 函数 | 描述 |
| --- | --- |
| `int dayOfYear(DateTime myDateTime)` | 获取 DateTime 值的一年中的某天 (用整数表示)。<br><br>示例：`#{dayOfYear(myDateTime)}`<br><br>结果：`144` |
| `DateTime firstOfMonth(DateTime myDateTime)` | 使用指定的 DateTime 为月的开始时间创建 DateTime 对象。<br><br>示例：`#{firstOfMonth(myDateTime)}`<br><br>结果：`"2011-05-01T17:10:00z"` |
| `String format(DateTime myDateTime,String format)` | 创建一个字符串对象，它是使用指定的格式字符串转换指定 DateTime 的结果。<br><br>示例：`#{format(myDateTime,'YYYY-MM-dd HH:mm:ss z')}`<br><br>结果：`"2011-05-24T17:10:00 UTC"` |
| `int hour(DateTime myDateTime)` | 获取 DateTime 值的小时 (用整数表示)。<br><br>示例：`#{hour(myDateTime)}`<br><br>结果：`17` |

| 函数 | 描述 |
| --- | --- |
| `DateTime makeDate(int year,int month,int day)` | 使用指定的年、月和日为午夜创建一个 DateTime 对象 (用 UTC 表示)。<br><br>示例：`#{makeDate(2011,5,24)}`<br><br>结果：`"2011-05-24T0:00:00z"` |
| `DateTime makeDateTime(int year,int month,int day,int hour,int minute)` | 使用指定的年、月、日、小时和分钟创建一个 DateTime 对象 (用 UTC 表示)。<br><br>示例：`#{makeDateTime(2011,5,24,14,21)}`<br><br>结果：`"2011-05-24T14:21:00z"` |
| `DateTime midnight(DateTime myDateTime)` | 为当前午夜创建一个相对于指定 DateTime 的 DateTime 对象。例如，当 MyDateTime 为 2011-05-25T17:10:00z 时，结果如下所示。<br><br>示例：`#{midnight(myDateTime)}`<br><br>结果：`"2011-05-25T0:00:00z"` |

| 函数 | 描述 |
| --- | --- |
| DateTime minusDays(DateTime myDateTime,int daysToSub) | 创建一个 DateTime 对象，它是从指定 DateTime 中减去指定天数的结果。<br><br>示例：#{minusDays(myDateTime,1)}<br><br>结果："2011-05-23T17:10:00z" |
| DateTime minusHours(DateTime myDateTime,int hoursToSub) | 创建一个 DateTime 对象，它是从指定 DateTime 中减去指定小时数的结果。<br><br>示例：#{minusHours(myDateTime,1)}<br><br>结果："2011-05-24T16:10:00z" |
| DateTime minusMinutes(DateTime myDateTime,int minutesToSub) | 创建一个 DateTime 对象，它是从指定 DateTime 中减去指定分钟数的结果。<br><br>示例：#{minusMinutes(myDateTime,1)}<br><br>结果："2011-05-24T17:09:00z" |

| 函数 | 描述 |
| --- | --- |
| `DateTime minusMonths(DateTime myDateTime,int monthsToSub)` | 创建一个 DateTime 对象，它是从指定 DateTime 中减去指定月数的结果。<br><br>示例：`#{minusMonths(myDateTime,1)}`<br><br>结果："2011-04-24T17:10:00z" |
| `DateTime minusWeeks(DateTime myDateTime,int weeksToSub)` | 创建一个 DateTime 对象，它是从指定 DateTime 中减去指定星期数的结果。<br><br>示例：`#{minusWeeks(myDateTime,1)}`<br><br>结果："2011-05-17T17:10:00z" |
| `DateTime minusYears(DateTime myDateTime,int yearsToSub)` | 创建一个 DateTime 对象，它是从指定 DateTime 中减去指定年数的结果。<br><br>示例：`#{minusYears(myDateTime,1)}`<br><br>结果："2010-05-24T17:10:00z" |
| `int minute(DateTime myDateTime)` | 获取 DateTime 值的分钟 (用整数表示)。<br><br>示例：`#{minute(myDateTime)}`<br><br>结果：10 |

| 函数 | 描述 |
| --- | --- |
| `int month(DateTime myDateTime)` | 获取 DateTime 值的月 (用整数表示)。<br><br>示例：`#{month(myDateTime)}`<br><br>结果：`5` |
| `DateTime plusDays(DateTime myDateTime,int daysToAdd)` | 创建一个 DateTime 对象，它是将指定天数加上指定 DateTime 的结果。<br><br>示例：`#{plusDays(myDateTime,1)}`<br><br>结果：`"2011-05-25T17:10:00z"` |
| `DateTime plusHours(DateTime myDateTime,int hoursToAdd)` | 创建一个 DateTime 对象，它是将指定小时数加上指定 DateTime 的结果。<br><br>示例：`#{plusHours(myDateTime,1)}`<br><br>结果：`"2011-05-24T18:10:00z"` |
| `DateTime plusMinutes(DateTime myDateTime,int minutesToAdd)` | 创建一个 DateTime 对象，它是将指定分钟数加上指定 DateTime 的结果。<br><br>示例：`#{plusMinutes(myDateTime,1)}`<br><br>结果：`"2011-05-24 17:11:00z"` |

| 函数 | 描述 |
| --- | --- |
| DateTime plusMonths(DateTime myDateTime,int monthsToAdd) | 创建一个 DateTime 对象，它是将指定月数加上指定 DateTime 的结果。<br><br>示例：#{plusMonths(myDateTime,1)}<br><br>结果："2011-06-24T17:10:00z" |
| DateTime plusWeeks(DateTime myDateTime,int weeksToAdd) | 创建一个 DateTime 对象，它是将指定星期数加上指定 DateTime 的结果。<br><br>示例：#{plusWeeks(myDateTime,1)}<br><br>结果："2011-05-31T17:10:00z" |
| DateTime plusYears(DateTime myDateTime,int yearsToAdd) | 创建一个 DateTime 对象，它是将指定年数加上指定 DateTime 的结果。<br><br>示例：#{plusYears(myDateTime,1)}<br><br>结果："2012-05-24T17:10:00z" |

| 函数 | 描述 |
| --- | --- |
| `DateTime sunday(DateTime myDateTime)` | 为上一个星期日创建一个相对于指定的 DateTime 的 DateTime 对象。如果指定的 DateTime 为星期日，则结果为指定的 DateTime。<br><br>示例：`#{sunday(myDateTime)}`<br><br>结果：`"2011-05-22 17:10:00 UTC"` |
| `int year(DateTime myDateTime)` | 获取 DateTime 值的年 (用整数表示)。<br><br>示例：`#{year(myDateTime)}`<br><br>结果：`2011` |
| `DateTime yesterday(DateTime myDateTime)` | 为上一天创建一个相对于指定的 DateTime 的 DateTime 对象。结果与 minusDays(1) 的相同。<br><br>示例：`#{yesterday(myDateTime)}`<br><br>结果：`"2011-05-23T17:10:00z"` |

# 特殊字符

AWS Data Pipeline 在管道定义中使用具有特殊含义的特定字符，如下表所示。

| 特殊字符 | 描述 | 示例 |
| --- | --- | --- |
| @ | 运行时字段。此字符是仅在管道运行时可用的字段的字段名前缀。 | @actualStartTime<br><br>@failureReason<br><br>@resourceStatus |
| # | 表达式。表达式用“#{”和“}”分隔，大括号内的内容由 AWS Data Pipeline 计算。有关更多信息，请参阅表达式。 | #{format(myDateTime,'YYYY-MM-dd hh:mm:ss')}<br><br>s3://mybucket/#{id}.csv |
| * | 加密的字段。此字符是一个字段名称前缀，用于指示 AWS Data Pipeline 将对在控制台或 CLI 与 AWS Data Pipeline 服务之间传输的此字段内容进行加密。 | *password |

# 管道对象引用

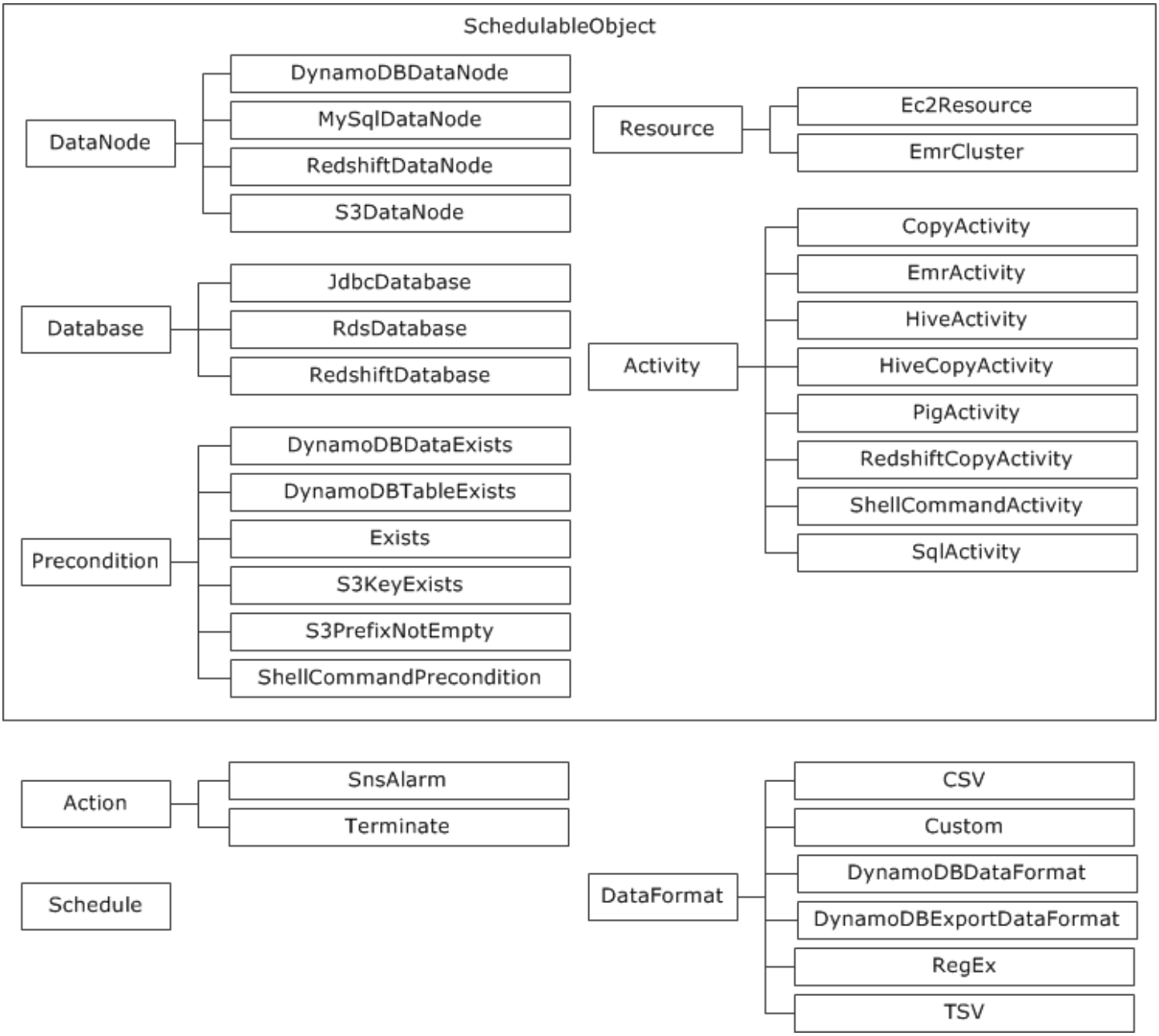
您可在管道定义文件中使用以下管道对象和组件。

内容



> **Note**
>
> 有关使用 AWS Data Pipeline Java 软件开发工具包的示例应用程序，请参阅 GitHub 上的 Data Pipeline DynamoDB 导出 Java 示例。

以下是 AWS Data Pipeline 的对象层次结构。




# 数据节点

以下是 AWS Data Pipeline 数据节点对象：

Objects

- DynamoDBDataNode
- MySqlDataNode
- RedshiftDataNode

- S3DataNode
- SqlDataNode

## DynamoDBDataNode

使用 DynamoDB 定义一个数据节点，该节点指定为 HiveActivity 或 EMRActivity 对象的输入。

> **Note**
>
> DynamoDBDataNode 对象不支持 Exists 先决条件。

### 示例

以下是该对象类型的示例。该对象引用您在同一管道定义文件中定义的其他两个对象。CopyPeriod 为 Schedule 对象，Ready 为先决条件对象。

```
{
  "id" : "MyDynamoDBTable",
  "type" : "DynamoDBDataNode",
  "schedule" : { "ref" : "CopyPeriod" },
  "tableName" : "adEvents",
  "precondition" : { "ref" : "Ready" }
}
```

### 语法

| 必填字段 | 描述 | 槽类型 |
| --- | --- | --- |
| tableName | DynamoDB 表。 | String |

| 对象调用字段 | 描述 | 槽类型 |
| --- | --- | --- |
| 计划 | 该对象在计划间隔的执行中调用。用户必须指定对另一个对象的计划引用，以便设置该对象的依赖项执行顺序。用户可在对象上明确设置计划来满足此要求，例如，指定 "schedule": {"ref": | 引用对象，例如，"schedule":{"ref":"mySchedulelId"} |

| 对象调用字段 | 描述 | 槽类型 |
| --- | --- | --- |
| | "DefaultSchedule"}。在大多数情况下，最好将计划引用放在默认管道对象上，以便所有对象继承该计划。或者，如果管道有一个计划树 (计划位于主计划中)，用户可以创建具有计划引用的父对象。有关示例可选计划配置的更多信息，请参阅计划。 | |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| attemptStatus | 来自远程活动的最近报告的状态。 | String |
| attemptTimeout | 远程工作完成的超时时间。如果设置该字段，则可能会重试未在设置的开始时间内完成的远程活动。 | 周期 |
| dataFormat | 该数据节点所描述数据的 DataFormat。目前支持 HiveActivity 和 HiveCopyActivity。 | 引用对象，"dataFormat":{"ref":"myDynamoDBDataFormatId"} |
| dependsOn | 指定与另一个可运行对象的依赖关系 | 引用对象，例如，"dependsOn":{"ref":"myActivityId"} |
| failureAndRerunMode | 描述依赖项失败或重新运行时的使用者节点行为。 | 枚举 |
| lateAfterTimeout | 管道启动后经过的时间，在此时间内，对象必须完成。仅当计划类型未设置为 ondemand 时才会触发。 | 周期 |
| maxActiveInstances | 组件的并发活动实例的最大数量。重新运行不计入活动实例数中。 | 整数 |
| maximumRetries | 失败后的最大重试次数 | 整数 |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| onFail | 当前对象失败时要运行的操作。 | 引用对象，例如，"onFail":{"ref":"myActionId"} |
| onLateAction | 在尚未计划对象或对象仍未完成的情况下将触发的操作。 | 引用对象，例如，"onLateAction":{"ref":"myActionId"} |
| onSuccess | 当前对象成功时要运行的操作。 | 引用对象，例如，"onSuccess":{"ref":"myActionId"} |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |
| pipelineLogUri | 用于上传管道日志的 S3 URI (例如“s3://BucketName/Key/”)。 | String |
| precondition | (可选) 定义先决条件。在满足所有先决条件之前，数据节点不会标记为“READY”。 | 引用对象，例如，"precondition":{"ref":"myPreconditionId"} |
| readThroughputPercent | 设置读取操作的速率，将 DynamoDB 预配的吞吐速率保持在为您的表分配的范围内。该值为介于 0.1 到 1.0 之间的双精度值 (包含端点)。 | Double |
| region | DynamoDB 表所在区域的代码。例如，us-east-1。这由 HiveActivity 在 Hive 中为 DynamoDB 表执行暂存时使用。 | 枚举 |
| reportProgressTimeout | 远程工作对 reportProgress 的连续调用的超时时间。如果设置此字段，则未报告指定时段的进度的远程活动可能会被视为停滞且已重试。 | 周期 |
| retryDelay | 两次重试之间的超时时间。 | 周期 |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| runsOn | 运行活动或命令的计算资源。例如，Amazon EC2 实例或 Amazon EMR 集群。 | 引用对象，例如，"runsOn":{"ref":"myResourceId"} |
| scheduleType | 计划类型允许您指定应在间隔的结尾还是开头计划您管道定义中的对象。时间序列风格计划表示在每次间隔的结尾计划实例，而 Cron 风格计划表示应在每次间隔的开头计划实例。按需计划让您可以在每次激活时运行一次管道。这意味着，您不需要克隆或重新创建管道以再次运行它。如果您使用按需计划，则必须在默认对象中指定它，并且必须是在管道中为对象指定的唯一 scheduleType。要使用按需管道，您只需为每个后续运行调用 ActivatePipeline 操作。值包括：cron、ondemand 和 timeseries。 | 枚举 |
| workerGroup | 工作线程组。这可用于路由任务。如果您提供 runsOn 值并且存在 workerGroup，则将忽略 workerGroup。 | String |
| writeThroughputPercent | 设置写入操作的速率，将 DynamoDB 预配的吞吐速率保持在为您的表分配的范围内。该值为介于 0.1 到 1.0 之间的双精度值 (包含端点)。 | Double |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @activeInstances | 当前计划的有效实例对象的列表。 | 引用对象，例如，"activeInstances":{"ref":"myRunnableObjectId"} |
| @actualEndTime | 该对象的执行完成时间。 | DateTime |
| @actualStartTime | 该对象的执行开始时间。 | DateTime |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| cancellationReason | 该对象被取消时显示的 cancellationReason。 | String |
| @cascadeFailedOn | 对象在其上失败的依赖项链的描述。 | 引用对象，例如，"cascadeFailedOn":{"ref":"myRunnableObjectId"} |
| emrStepLog | 仅在尝试 EMR 活动时可用的 EMR 步骤日志 | String |
| errorId | 该对象失败时显示的 errorId。 | String |
| errorMessage | 该对象失败时显示的 errorMessage。 | String |
| errorStackTrace | 该对象失败时显示的错误堆栈跟踪。 | String |
| @finishedTime | 该对象完成其执行的时间。 | DateTime |
| hadoopJobLog | 在尝试基于 EMR 的活动时可用的 Hadoop 任务日志。 | String |
| @healthStatus | 对象的运行状况，反映进入终止状态的上个对象实例成功还是失败。 | String |
| @healthStatusFromInstanceId | 进入终止状态的上个实例对象的 ID。 | String |
| @healthStatusUpdatedTime | 上次更新运行状况的时间。 | DateTime |
| hostname | 已执行任务尝试的客户端的主机名。 | String |
| @lastDeactivatedTime | 上次停用该对象的时间。 | DateTime |
| @latestCompletedRunTime | 已完成执行的最新运行的时间。 | DateTime |
| @latestRunTime | 已计划执行的最新运行的时间。 | DateTime |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @nextRunTime | 计划下次运行的时间。 | DateTime |
| reportProgressTime | 远程活动报告进度的最近时间。 | DateTime |
| @scheduledEndTime | 对象的计划结束时间。 | DateTime |
| @scheduledStartTime | 对象的计划开始时间。 | DateTime |
| @status | 该对象的状态。 | String |
| @version | 用来创建对象的管道版本。 | String |
| @waitingOn | 该对象在其上处于等待状态的依赖项列表的描述。 | 引用对象，例如，"waitingOn":{"ref": "myRunnableObjectId"} |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息 | String |
| @pipelineId | 该对象所属的管道的 ID | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象 | String |

## MySqlDataNode

使用 MySQL 定义数据节点。

> **Note**
>
> `MySqlDataNode` 类型已被弃用。建议您改用 SqlDataNode。

## 示例

以下是该对象类型的示例。该对象引用您在同一管道定义文件中定义的其他两个对象。CopyPeriod 为 Schedule 对象，Ready 为先决条件对象。

```
{
  "id" : "Sql Table",
  "type" : "MySqlDataNode",
  "schedule" : { "ref" : "CopyPeriod" },
  "table" : "adEvents",
  "username": "user_name",
  "*password": "my_password",
  "connectionString": "jdbc:mysql://mysqlinstance-rds.example.us-
east-1.rds.amazonaws.com:3306/database_name",
  "selectQuery" : "select * from #{table} where eventTime >=
 '#{@scheduledStartTime.format('YYYY-MM-dd HH:mm:ss')}' and eventTime <
 '#{@scheduledEndTime.format('YYYY-MM-dd HH:mm:ss')}'",
  "precondition" : { "ref" : "Ready" }
}
```

### 语法

| 必填字段 | 描述 | 槽类型 |
| --- | --- | --- |
| table | MySQL 数据库中的表的名称。 | String |

| 对象调用字段 | 描述 | 槽类型 |
| --- | --- | --- |
| 计划 | 该对象在计划间隔的执行中调用。用户必须指定对另一个对象的计划引用，以便设置该对象的依赖项执行顺序。用户可在对象上明确设置计划来满足此要求，例如，指定 "schedule": {"ref": "DefaultSchedule"}。在大多数情况下，最好将计划引用放在默认管道对象上，以便所有对象继承该计划。或者，如果管道有一个计划树 (计划位于主计划中)，用户可以创建具有计划引用的父对象。有关示例可选计划配置的更 | 引用对象，例如，"schedule":{"ref":"mScheduleId"} |

| 对象调用字段 | 描述 | 槽类型 |
| --- | --- | --- |
| | 多信息，请参阅 https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html。 | |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| attemptStatus | 来自远程活动的最近报告的状态。 | String |
| attemptTimeout | 远程工作完成的超时时间。如果设置此字段，则可能会重试未在设定的开始时间内完成的远程活动。 | 周期 |
| createTableSql | SQL 创建表表达式，该表达式创建表。 | String |
| 数据库 | 数据库的名称。 | 引用对象，例如，"database":{"ref":"myDatabaseId"} |
| dependsOn | 指定与另一个可运行对象的依赖关系。 | 引用对象，例如，"dependsOn":{"ref":"myActivityId"} |
| failureAndRerunMode | 描述依赖项失败或重新运行时的使用者节点行为。 | 枚举 |
| insertQuery | 将数据插入到表中的 SQL 语句。 | String |
| lateAfterTimeout | 管道启动后经过的时间，在此时间内，对象必须完成。仅当计划类型未设置为 ondemand 时才会触发。 | 周期 |
| maxActiveInstances | 组件的并发活动实例的最大数量。重新运行不计入活动实例数中。 | 整数 |
| maximumRetries | 失败后的最大重试次数 | 整数 |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| onFail | 当前对象失败时要运行的操作。 | 引用对象，例如，"onFail":{"ref":"myActionId"} |
| onLateAction | 在尚未计划对象或对象仍未完成的情况下将触发的操作。 | 引用对象，例如，"onLateAction":{"ref":"myActionId"} |
| onSuccess | 当前对象成功时要运行的操作。 | 引用对象，例如，"onSuccess":{"ref":"myActionId"} |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |
| pipelineLogUri | 用于上传管道日志的 S3 URI (例如“s3://BucketName/Key/”)。 | String |
| precondition | (可选) 定义先决条件。在满足所有先决条件之前，数据节点不会标记为“READY”。 | 引用对象，例如，"precondition":{"ref":"myPreconditionId"} |
| reportProgressTimeout | 远程工作对 reportProgress 的连续调用的超时时间。如果设置此字段，则未报告指定时段的进度的远程活动可能会被视为停滞且已重试。 | 周期 |
| retryDelay | 两次重试之间的超时时间。 | 周期 |
| runsOn | 运行活动或命令的计算资源。例如，Amazon EC2 实例或 Amazon EMR 集群。 | 引用对象，例如，"runsOn":{"ref":"myResourceId"} |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| scheduleType | 计划类型允许您指定应在间隔的结尾还是开头计划您管道定义中的对象。时间序列风格计划表示在每次间隔的结尾计划实例，而 Cron 风格计划表示应在每次间隔的开头计划实例。按需计划让您可以在每次激活时运行一次管道。这意味着，您不需要克隆或重新创建管道以再次运行它。如果您使用按需计划，则必须在默认对象中指定它，并且必须是在管道中为对象指定的唯一 scheduleType。要使用按需管道，您只需为每个后续运行调用 ActivatePipeline 操作。值包括：cron、ondemand 和 timeseries。 | 枚举 |
| schemaName | 保存表的架构的名称 | String |
| selectQuery | 从表中提取数据的 SQL 语句。 | String |
| workerGroup | 工作线程组。这可用于路由任务。如果您提供 runsOn 值并且存在 workerGroup，则将忽略 workerGroup。 | String |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @activeInstances | 当前计划的有效实例对象的列表。 | 引用对象，例如，"activeInstances":{"ref":"myRunnableObjectId"} |
| @actualEndTime | 该对象的执行完成时间。 | DateTime |
| @actualStartTime | 该对象的执行开始时间。 | DateTime |
| cancellationReason | 该对象被取消时显示的 cancellationReason。 | String |
| @cascadeFailedOn | 对象在其上失败的依赖项链的描述。 | 引用对象，例如，"cascadeFailedOn":{ |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| | | "ref":"myRunnableObjectId"} |
| emrStepLog | 仅在尝试 EMR 活动时可用的 EMR 步骤日志 | String |
| errorId | 该对象失败时显示的 errorId。 | String |
| errorMessage | 该对象失败时显示的 errorMessage。 | String |
| errorStackTrace | 该对象失败时显示的错误堆栈跟踪。 | String |
| @finishedTime | 该对象完成其执行的时间。 | DateTime |
| hadoopJobLog | 在尝试基于 EMR 的活动时可用的 Hadoop 任务日志。 | String |
| @healthStatus | 对象的运行状况，反映进入终止状态的上个对象实例成功还是失败。 | String |
| @healthStatusFromInstanceId | 进入终止状态的上个实例对象的 ID。 | String |
| @healthStatusUpdatedTime | 上次更新运行状况的时间。 | DateTime |
| hostname | 已执行任务尝试的客户端的主机名。 | String |
| @lastDeactivatedTime | 上次停用该对象的时间。 | DateTime |
| @latestCompletedRunTime | 已完成执行的最新运行的时间。 | DateTime |
| @latestRunTime | 已计划执行的最新运行的时间。 | DateTime |
| @nextRunTime | 计划下次运行的时间。 | DateTime |
| reportProgressTime | 远程活动报告进度的最近时间。 | DateTime |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @scheduledEndTime | 对象的计划结束时间。 | DateTime |
| @scheduledStartTime | 对象的计划开始时间。 | DateTime |
| @status | 该对象的状态。 | String |
| @version | 用来创建对象的管道版本。 | String |
| @waitingOn | 该对象在其上处于等待状态的依赖项列表的描述。 | 引用对象，例如，"waitingOn":{"ref": "myRunnableObjectId"} |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息。 | String |
| @pipelineId | 该对象所属的管道的 ID。 | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象。 | String |

### 另请参阅

- S3DataNode

## RedshiftDataNode

使用 Amazon Redshift 定义一个数据节点。RedshiftDataNode 表示由管道使用的数据库中的数据（如数据表）的属性。

### 示例

以下是该对象类型的示例。

```
{
  "id" : "MyRedshiftDataNode",
  "type" : "RedshiftDataNode",
  "database": { "ref": "MyRedshiftDatabase" },
  "tableName": "adEvents",
  "schedule": { "ref": "Hour" }
}
```

## 语法

| 必填字段 | 描述 | 槽类型 |
| --- | --- | --- |
| 数据库 | 表所在的数据库。 | 引用对象，例如，"database":{"ref":"myRedshiftDatabaseId"} |
| tableName | Amazon Redshift 表的名称。如果此表不存在并且您已提供 createTableSql，则将创建此表。 | String |

| 对象调用字段 | 描述 | 槽类型 |
| --- | --- | --- |
| 计划 | 该对象在计划间隔的执行中调用。用户必须指定对另一个对象的计划引用，以便设置该对象的依赖项执行顺序。用户可在对象上明确设置计划来满足此要求，例如，指定 "schedule": {"ref": "DefaultSchedule"}。在大多数情况下，最好将计划引用放在默认管道对象上，以便所有对象继承该计划。或者，如果管道有一个计划树 (计划位于主计划中)，用户可以创建具有计划引用的父对象。有关示例可选计划配置的更多信息，请参阅 https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html。 | 引用对象，例如，"schedule":{"ref":"myScheduleId"} |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| attemptStatus | 来自远程活动的最近报告的状态。 | String |
| attemptTimeout | 远程工作完成的超时时间。如果设置此字段，则可能会重试未在设定的开始时间内完成的远程活动。 | 周期 |
| createTableSql | 用于在数据库中创建表的 SQL 表达式。我们建议您指定应在其中创建表的架构，例如：CREATE TABLE mySchema.myTable (bestColumn varchar(25) primary key distkey, numberOfWins integer sortKey)。如果由 tableName 指定的表在 schemaName 字段所指定的架构中不存在，则 AWS Data Pipeline 将在 createTableSql 字段中运行脚本。例如，如果您指定 schemaName 作为 mySchema，但未在 createTableSql 字段中包含 mySchema，则将在错误的架构中创建表 (默认情况下，将在 PUBLIC 中创建表)。由于 AWS Data Pipeline 无法分析您的 CREATE TABLE 语句，因此将发生此情况。 | String |
| dependsOn | 指定与另一个可运行对象的依赖关系 | 引用对象，例如，"dependsOn":{"ref":"myActivityId"} |
| failureAndRerunMode | 描述依赖项失败或重新运行时的使用者节点行为。 | 枚举 |
| lateAfterTimeout | 管道启动后经过的时间，在此时间内，对象必须完成。仅当计划类型未设置为 ondemand 时才会触发。 | 周期 |
| maxActiveInstances | 组件的并发活动实例的最大数量。重新运行不计入活动实例数中。 | 整数 |
| maximumRetries | 失败后的最大重试次数。 | 整数 |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| onFail | 当前对象失败时要运行的操作。 | 引用对象，例如，"onFail":{"ref":"myActionId"} |
| onLateAction | 在尚未计划对象或对象仍未完成的情况下将触发的操作。 | 引用对象，例如，"onLateAction":{"ref":"myActionId"} |
| onSuccess | 当前对象成功时要运行的操作。 | 引用对象，例如，"onSuccess":{"ref":"myActionId"} |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |
| pipelineLogUri | 用于上传管道日志的 S3 URI (例如“s3://BucketName/Key/”)。 | String |
| precondition | (可选) 定义先决条件。在满足所有先决条件之前，数据节点不会标记为“READY”。 | 引用对象，例如，"precondition":{"ref":"myPreconditionId"} |
| primaryKeys | 如果您未在 `RedShiftCopyActivity` 中为目标表指定 primaryKeys，则可使用将用作 mergeKey 的 primaryKeys 指定列的列表。不过，如果您现已在 Amazon Redshift 表中定义一个 primaryKey，此设置将覆盖现有密钥。 | String |
| reportProgressTimeout | 远程工作对 reportProgress 的连续调用的超时时间。如果设置此字段，则未报告指定时段的进度的远程活动可能会被视为停滞且已重试。 | 周期 |
| retryDelay | 两次重试之间的超时时间。 | 周期 |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| runsOn | 运行活动或命令的计算资源。例如，Amazon EC2 实例或 Amazon EMR 集群。 | 引用对象，例如，"runsOn":{"ref":"myResourceId"} |
| scheduleType | 计划类型允许您指定应在间隔的结尾还是开头计划您管道定义中的对象。时间序列风格计划表示在每次间隔的结尾计划实例，而 Cron 风格计划表示应在每次间隔的开头计划实例。按需计划让您可以在每次激活时运行一次管道。这意味着，您不需要克隆或重新创建管道以再次运行它。如果您使用按需计划，则必须在默认对象中指定它，并且必须是在管道中为对象指定的唯一 scheduleType。要使用按需管道，您只需为每个后续运行调用 ActivatePipeline 操作。值包括：cron、ondemand 和 timeseries。 | 枚举 |
| schemaName | 此可选字段指定 Amazon Redshift 表的架构的名称。如果未指定架构名称，则架构名称为 PUBLIC，这是在 Amazon Redshift 中的默认架构。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》。 | String |
| workerGroup | 工作线程组。这可用于路由任务。如果您提供 runsOn 值并且存在 workerGroup，则将忽略 workerGroup。 | String |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @activeInstances | 当前计划的有效实例对象的列表。 | 引用对象，例如，"activeInstances":{"ref":"myRunnableObjectId"} |
| @actualEndTime | 该对象的执行完成时间。 | DateTime |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @actualStartTime | 该对象的执行开始时间。 | DateTime |
| cancellationReason | 该对象被取消时显示的 cancellationReason。 | String |
| @cascadeFailedOn | 对象在其上失败的依赖项链的描述。 | 引用对象，例如，"cascadeFailedOn":{"ref":"myRunnableObjectId"} |
| emrStepLog | 仅在尝试 EMR 活动时可用的 EMR 步骤日志 | String |
| errorId | 该对象失败时显示的 errorId。 | String |
| errorMessage | 该对象失败时显示的 errorMessage。 | String |
| errorStackTrace | 该对象失败时显示的错误堆栈跟踪。 | String |
| @finishedTime | 该对象完成其执行的时间。 | DateTime |
| hadoopJobLog | 在尝试基于 EMR 的活动时可用的 Hadoop 任务日志。 | String |
| @healthStatus | 对象的运行状况，反映进入终止状态的上个对象实例成功还是失败。 | String |
| @healthStatusFromInstanceId | 进入终止状态的上个实例对象的 ID。 | String |
| @healthStatusUpdatedTime | 上次更新运行状况的时间。 | DateTime |
| hostname | 已执行任务尝试的客户端的主机名。 | String |
| @lastDeactivatedTime | 上次停用该对象的时间。 | DateTime |
| @latestCompletedRunTime | 已完成执行的最新运行的时间。 | DateTime |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @latestRunTime | 已计划执行的最新运行的时间。 | DateTime |
| @nextRunTime | 计划下次运行的时间。 | DateTime |
| reportProgressTime | 远程活动报告进度的最近时间。 | DateTime |
| @scheduledEndTime | 对象的计划结束时间。 | DateTime |
| @scheduledStartTime | 对象的计划开始时间。 | DateTime |
| @status | 该对象的状态。 | String |
| @version | 用来创建对象的管道版本。 | String |
| @waitingOn | 该对象在其上处于等待状态的依赖项列表的描述。 | 引用对象，例如，"waitingOn":{"ref": "myRunnableObjectId"} |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息。 | String |
| @pipelineId | 该对象所属的管道的 ID。 | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象。 | String |

## S3DataNode

使用 Amazon S3 定义数据节点。默认情况下，S3DataNode 使用服务器端加密。如果您想禁止此功能，可将 s3EncryptionType 设置为 NONE。

> **Note**
>
> 在使用 S3DataNode 作为 CopyActivity 的输入时，仅 CSV 和 TSV 数据格式受支持。

## 示例

以下是该对象类型的示例。该对象引用您在同一管道定义文件中定义的另一个对象。CopyPeriod 为 Schedule 对象。

```
{
  "id" : "OutputData",
  "type" : "S3DataNode",
  "schedule" : { "ref" : "CopyPeriod" },
  "filePath" : "s3://myBucket/#{@scheduledStartTime}.csv"
}
```

## 语法

| 对象调用字段 | 描述 | 槽类型 |
| --- | --- | --- |
| 计划 | 该对象在计划间隔的执行中调用。用户必须指定对另一个对象的计划引用，以便设置该对象的依赖项执行顺序。用户可在对象上明确设置计划来满足此要求，例如，指定 "schedule": {"ref": "DefaultSchedule"}。在大多数情况下，最好将计划引用放在默认管道对象上，以便所有对象继承该计划。或者，如果管道有一个计划树 (计划位于主计划中)，用户可以创建具有计划引用的父对象。有关示例可选计划配置的更多信息，请参阅 https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html。 | 引用对象，例如，"schedule":{"ref":"myScheduleId"} |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| attemptStatus | 来自远程活动的最近报告的状态。 | String |
| attemptTimeout | 远程工作完成的超时时间。如果设置此字段，则可能会重试未在设定的开始时间内完成的远程活动。 | 周期 |
| compression | S3DataNode 所描述数据的压缩类型。“none”表示未压缩，“gzip”是使用 gzip 算法压缩。此字段仅支持用于 Amazon Redshift 并将 S3DataNode 与 CopyActivity 配合使用时。 | 枚举 |
| dataFormat | 此 S3DataNode 所描述数据的 DataFormat。 | 引用对象，例如，"dataFormat":{"ref":"myDataFormatId"} |
| dependsOn | 指定与另一个可运行对象的依赖关系 | 引用对象，例如，"dependsOn":{"ref":"myActivityId"} |
| directoryPath | 将 Amazon S3 目录路径作为 URI：s3://my-bucket/my-key-for-directory。您必须提供 filePath 或 directoryPath 值。 | String |
| failureAndRerunMode | 描述依赖项失败或重新运行时的使用者节点行为。 | 枚举 |
| filePath | 将 Amazon S3 中对象的路径作为 URI，例如：s3://my-bucket/my-key-for-file。您必须提供 filePath 或 directoryPath 值。它们表示一个文件夹和文件名。使用 directoryPath 值适应目录中的多个文件。 | String |
| lateAfterTimeout | 管道启动后经过的时间，在此时间内，对象必须完成。仅当计划类型未设置为 ondemand 时才会触发。 | 周期 |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| manifestFilePath | 使用 Amazon Redshift 支持格式的清单文件的 Amazon S3 路径。AWS Data Pipeline 使用该清单文件将指定的 Amazon S3 文件复制到表中。此字段仅在 RedShiftCopyActivity 引用 S3DataNode 时有效。 | String |
| maxActiveInstances | 组件的并发活动实例的最大数量。重新运行不计入活动实例数中。 | 整数 |
| maximumRetries | 失败后的最大重试次数 | 整数 |
| onFail | 当前对象失败时要运行的操作。 | 引用对象，例如，"onFail":{"ref":"myActionId"} |
| onLateAction | 在尚未计划对象或对象仍未完成的情况下将触发的操作。 | 引用对象，例如，"onLateAction":{"ref":"myActionId"} |
| onSuccess | 当前对象成功时要运行的操作。 | 引用对象，例如，"onSuccess":{"ref":"myActionId"} |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |
| pipelineLogUri | 用于上传管道日志的 S3 URI (例如“s3://BucketName/Key/”)。 | String |
| precondition | (可选) 定义先决条件。在满足所有先决条件之前，数据节点不会标记为“READY”。 | 引用对象，例如，"precondition":{"ref":"myPreconditionId"} |
| reportProgressTimeout | 远程工作对 reportProgress 的连续调用的超时时间。如果设置此字段，则未报告指定时段的进度的远程活动可能会被视为停滞且已重试。 | 周期 |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| retryDelay | 两次重试之间的超时时间。 | 周期 |
| runsOn | 运行活动或命令的计算资源。例如，Amazon EC2 实例或 Amazon EMR 集群。 | 引用对象，例如，"runsOn":{"ref":"myResourceId"} |
| s3EncryptionType | 覆盖 Amazon S3 加密类型。值为 SERVER_SIDE_ENCRYPTION 或 NONE。默认情况下启用服务器端加密。 | 枚举 |
| scheduleType | 计划类型允许您指定应在间隔的结尾还是开头计划您管道定义中的对象。时间序列风格计划表示在每次间隔的结尾计划实例，而 Cron 风格计划表示应在每次间隔的开头计划实例。按需计划让您可以在每次激活时运行一次管道。这意味着，您不需要克隆或重新创建管道以再次运行它。如果您使用按需计划，则必须在默认对象中指定它，并且必须是在管道中为对象指定的唯一 scheduleType。要使用按需管道，您只需为每个后续运行调用 ActivatePipeline 操作。值包括：cron、ondemand 和 timeseries。 | 枚举 |
| workerGroup | 工作线程组。这可用于路由任务。如果您提供 runsOn 值并且存在 workerGroup，则将忽略 workerGroup。 | String |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @activeInstances | 当前计划的有效实例对象的列表。 | 引用对象，例如，"activeInstances":{"ref":"myRunnableObjectId"} |
| @actualEndTime | 该对象的执行完成时间。 | DateTime |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @actualStartTime | 该对象的执行开始时间。 | DateTime |
| cancellationReason | 该对象被取消时显示的 cancellationReason。 | String |
| @cascadeFailedOn | 对象在其上失败的依赖项链的描述。 | 引用对象，例如，"cascadeFailedOn":{"ref":"myRunnableObjectId"} |
| emrStepLog | 仅在尝试 EMR 活动时可用的 EMR 步骤日志 | String |
| errorId | 该对象失败时显示的 errorId。 | String |
| errorMessage | 该对象失败时显示的 errorMessage。 | String |
| errorStackTrace | 该对象失败时显示的错误堆栈跟踪。 | String |
| @finishedTime | 该对象完成其执行的时间。 | DateTime |
| hadoopJobLog | 在尝试基于 EMR 的活动时可用的 Hadoop 任务日志。 | String |
| @healthStatus | 对象的运行状况，反映进入终止状态的上个对象实例成功还是失败。 | String |
| @healthStatusFromInstanceId | 进入终止状态的上个实例对象的 ID。 | String |
| @healthStatusUpdatedTime | 上次更新运行状况的时间。 | DateTime |
| hostname | 已执行任务尝试的客户端的主机名。 | String |
| @lastDeactivatedTime | 上次停用该对象的时间。 | DateTime |
| @latestCompletedRunTime | 已完成执行的最新运行的时间。 | DateTime |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @latestRunTime | 已计划执行的最新运行的时间。 | DateTime |
| @nextRunTime | 计划下次运行的时间。 | DateTime |
| reportProgressTime | 远程活动报告进度的最近时间。 | DateTime |
| @scheduledEndTime | 对象的计划结束时间。 | DateTime |
| @scheduledStartTime | 对象的计划开始时间。 | DateTime |
| @status | 该对象的状态。 | String |
| @version | 用来创建对象的管道版本。 | String |
| @waitingOn | 该对象在其上处于等待状态的依赖项列表的描述。 | 引用对象，例如，"waitingOn":{"ref": "myRunnableObjectId"} |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息 | String |
| @pipelineId | 该对象所属的管道的 ID | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象 | String |

### 另请参阅

- MySqlDataNode

## SqlDataNode

使用 SQL 定义数据节点。

## 示例

以下是该对象类型的示例。该对象引用您在同一管道定义文件中定义的其他两个对象。CopyPeriod 为 Schedule 对象，Ready 为先决条件对象。

```
{
  "id" : "Sql Table",
  "type" : "SqlDataNode",
  "schedule" : { "ref" : "CopyPeriod" },
  "table" : "adEvents",
  "database":"myDataBaseName",
  "selectQuery" : "select * from #{table} where eventTime >=
 '#{@scheduledStartTime.format('YYYY-MM-dd HH:mm:ss')}' and eventTime <
 '#{@scheduledEndTime.format('YYYY-MM-dd HH:mm:ss')}'",
  "precondition" : { "ref" : "Ready" }
}
```

## 语法

| 必填字段 | 描述 | 槽类型 |
| --- | --- | --- |
| table | SQL 数据库中的表的名称。 | String |

| 对象调用字段 | 描述 | 槽类型 |
| --- | --- | --- |
| 计划 | 该对象在计划间隔的执行中调用。用户必须指定对另一个对象的计划引用，以便设置该对象的依赖项执行顺序。用户可在对象上明确设置计划来满足此要求，例如，指定 "schedule": {"ref": "DefaultSchedule"}。在大多数情况下，最好将计划引用放在默认管道对象上，以便所有对象继承该计划。或者，如果管道有一个计划树 (计划位于主计划中)，用户可以创建具有计划引用的父对象。有关示例可选计划配置的更多信息，请参阅 https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html。 | 引用对象，例如，"schedule":{"ref":"myScheduleId"} |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| attemptStatus | 来自远程活动的最近报告的状态。 | String |
| attemptTimeout | 远程工作完成的超时时间。如果设置此字段，则可能会重试未在设定的开始时间内完成的远程活动。 | 周期 |
| createTableSql | SQL 创建表表达式，该表达式创建表。 | String |
| 数据库 | 数据库的名称。 | 引用对象，例如，"database":{"ref":"myDatabaseId"} |
| dependsOn | 指定与另一个可运行对象的依赖关系。 | 引用对象，例如，"dependsOn":{"ref":"myActivityId"} |
| failureAndRerunMode | 描述依赖项失败或重新运行时的使用者节点行为。 | 枚举 |
| insertQuery | 将数据插入到表中的 SQL 语句。 | String |
| lateAfterTimeout | 管道启动后经过的时间，在此时间内，对象必须完成。仅当计划类型未设置为 ondemand 时才会触发。 | 周期 |
| maxActiveInstances | 组件的并发活动实例的最大数量。重新运行不计入活动实例数中。 | 整数 |
| maximumRetries | 失败后的最大重试次数 | 整数 |
| onFail | 当前对象失败时要运行的操作。 | 引用对象，例如，"onFail":{"ref":"myActionId"} |
| onLateAction | 在尚未计划对象或对象仍未完成的情况下将触发的操作。 | 引用对象，例如，"onLateAction":{"ref":"myActionId"} |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| onSuccess | 当前对象成功时要运行的操作。 | 引用对象，例如，"onSuccess":{"ref":"myActionId"} |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |
| pipelineLogUri | 用于上传管道日志的 S3 URI (例如“s3://BucketName/Key/”)。 | String |
| precondition | (可选) 定义先决条件。在满足所有先决条件之前，数据节点不会标记为“READY”。 | 引用对象，例如，"precondition":{"ref":"myPreconditionId"} |
| reportProgressTimeout | 远程工作对 reportProgress 的连续调用的超时时间。如果设置此字段，则未报告指定时段的进度的远程活动可能会被视为停滞且已重试。 | 周期 |
| retryDelay | 两次重试之间的超时时间。 | 周期 |
| runsOn | 运行活动或命令的计算资源。例如，Amazon EC2 实例或 Amazon EMR 集群。 | 引用对象，例如，"runsOn":{"ref":"myResourceId"} |
| scheduleType | 计划类型允许您指定应在间隔的结尾还是开头计划您管道定义中的对象。时间序列风格计划表示在每次间隔的结尾计划实例，而 Cron 风格计划表示应在每次间隔的开头计划实例。按需计划让您可以在每次激活时运行一次管道。这意味着，您不需要克隆或重新创建管道以再次运行它。如果您使用按需计划，则必须在默认对象中指定它，并且必须是在管道中为对象指定的唯一 scheduleType。要使用按需管道，您只需为每个后续运行调用 ActivatePipeline 操作。值包括：cron、ondemand 和 timeseries。 | 枚举 |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| schemaName | 保存表的架构的名称 | String |
| selectQuery | 从表中提取数据的 SQL 语句。 | String |
| workerGroup | 工作线程组。这可用于路由任务。如果您提供 runsOn 值并且存在 workerGroup，则将忽略 workerGroup。 | String |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @activeInstances | 当前计划的有效实例对象的列表。 | 引用对象，例如，"activeInstances":{"ref":"myRunnableObjectId"} |
| @actualEndTime | 该对象的执行完成时间。 | DateTime |
| @actualStartTime | 该对象的执行开始时间。 | DateTime |
| cancellationReason | 该对象被取消时显示的 cancellationReason。 | String |
| @cascadeFailedOn | 对象在其上失败的依赖项链的描述。 | 引用对象，例如，"cascadeFailedOn":{"ref":"myRunnableObjectId"} |
| emrStepLog | 仅在尝试 EMR 活动时可用的 EMR 步骤日志 | String |
| errorId | 该对象失败时显示的 errorId。 | String |
| errorMessage | 该对象失败时显示的 errorMessage。 | String |
| errorStackTrace | 该对象失败时显示的错误堆栈跟踪。 | String |
| @finishedTime | 该对象完成其执行的时间。 | DateTime |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| hadoopJobLog | 在尝试基于 EMR 的活动时可用的 Hadoop 任务日志。 | String |
| @healthStatus | 对象的运行状况，反映进入终止状态的上个对象实例成功还是失败。 | String |
| @healthStatusFromInstanceId | 进入终止状态的上个实例对象的 ID。 | String |
| @healthStatusUpdatedTime | 上次更新运行状况的时间。 | DateTime |
| hostname | 已执行任务尝试的客户端的主机名。 | String |
| @lastDeactivatedTime | 上次停用该对象的时间。 | DateTime |
| @latestCompletedRunTime | 已完成执行的最新运行的时间。 | DateTime |
| @latestRunTime | 已计划执行的最新运行的时间。 | DateTime |
| @nextRunTime | 计划下次运行的时间。 | DateTime |
| reportProgressTime | 远程活动报告进度的最近时间。 | DateTime |
| @scheduledEndTime | 对象的计划结束时间。 | DateTime |
| @scheduledStartTime | 对象的计划开始时间。 | DateTime |
| @status | 该对象的状态。 | String |
| @version | 用来创建对象的管道版本。 | String |
| @waitingOn | 该对象在其上处于等待状态的依赖项列表的描述。 | 引用对象，例如，"waitingOn":{"ref": "myRunnableObjectId"} |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息 | String |
| @pipelineId | 该对象所属的管道的 ID | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象 | String |

### 另请参阅

- S3DataNode

# 活动

以下是 AWS Data Pipeline 活动对象：

Objects

- CopyActivity
- EmrActivity
- HadoopActivity
- HiveActivity
- HiveCopyActivity
- PigActivity
- RedshiftCopyActivity
- ShellCommandActivity
- SqlActivity

## CopyActivity

将数据从一个位置复制到另一个位置。CopyActivity 支持 S3DataNode 和 SqlDataNode 作为输入和输出，并且通常会逐条记录执行复制操作。不过，当满足以下条件时，CopyActivity 可提供高性能的 Amazon S3 到 Amazon S3 复制：

- 输入和输出为 S3DataNodes
- 对于输入和输出，dataFormat 字段是相同的

如果您提供压缩的数据文件作为输入，并且不使用 S3 数据节点上的 compression 字段指示这一点，则 CopyActivity 可能会失败。在此情况下，CopyActivity 无法正确检测记录结束字符，并且操作会失败。此外，CopyActivity 支持从一个目录到另一个目录的复制以及将文件复制到目录中，但在将目录复制到文件时将执行逐条记录复制。最后，CopyActivity 不支持复制多部分 Amazon S3 文件。

CopyActivity 对其 CSV 支持有特定的限制。在使用 S3DataNode 作为 CopyActivity 的输入时，只能为 Amazon S3 输入和输出字段使用 CSV 数据文件格式的 Unix/Linux 变体。Unix/Linux 变体要求：

- 分隔符必须是“,”(逗号) 字符。
- 记录未用引号引起来。
- 默认转义字符是 ASCII 值 92 (反斜杠)。
- 记录结束标识符是 ASCII 值 10 (或“\n”)。

基于 Windows 的系统通常使用其他记录结束字符序列：回车符和换行符 (ASCII 值 13 和 ASCII 值 10)。您必须使用其他机制来适应此差异 (例如，使用预先复制脚本修改输入数据) 以确保 CopyActivity 能够正确检测到记录结尾；否则，CopyActivity 将反复失败。

在使用 CopyActivity 从 PostgreSQL RDS 对象导出到 TSV 数据格式时，默认 NULL 字符为 \n。

## 示例

以下是该对象类型的示例。该对象引用您将在同一管道定义文件中定义的其他三个对象。CopyPeriod 为 Schedule 对象，InputData 和 OutputData 为数据节点对象。

```
{
  "id" : "S3ToS3Copy",
  "type" : "CopyActivity",
  "schedule" : { "ref" : "CopyPeriod" },
  "input" : { "ref" : "InputData" },
  "output" : { "ref" : "OutputData" },
  "runsOn" : { "ref" : "MyEc2Resource" }
}
```

## 语法

| 对象调用字段 | 描述 | 槽类型 |
| --- | --- | --- |
| 计划 | 该对象在计划间隔的执行中调用。用户必须指定对另一个对象的计划引用，以便设置该对象的依赖项执行顺序。用户可在对象上明确设置计划来满足此要求，例如，指定 "schedule": {"ref": "DefaultSchedule"}。在大多数情况下，最好将计划引用放在默认管道对象上，以便所有对象继承该计划。或者，如果管道有一个计划树 (计划位于主计划中)，用户可以创建具有计划引用的父对象。有关示例可选计划配置的更多信息，请参阅 https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html。 | 引用对象，例如，"schedule":{"ref":"myScheduleId"} |

| 所需的组 (下列选项之一是必需的) | 描述 | 槽类型 |
| --- | --- | --- |
| runsOn | 运行活动或命令的计算资源。例如，Amazon EC2 实例或 Amazon EMR 集群。 | 引用对象，例如，"runsOn":{"ref":"myResourceId"} |
| workerGroup | 工作线程组。这可用于路由任务。如果您提供 runsOn 值并且存在 workerGroup，则将忽略 workerGroup。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| attemptStatus | 来自远程活动的最近报告的状态。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| attemptTimeout | 远程工作完成的超时时间。如果设置此字段，则可能会重试未在设定的开始时间内完成的远程活动。 | 周期 |
| dependsOn | 指定与另一个可运行对象的依赖关系。 | 引用对象，例如，"dependsOn":{"ref":"myActivityId"} |
| failureAndRerunMode | 描述依赖项失败或重新运行时的使用者节点行为。 | 枚举 |
| input | 输入数据源。 | 引用对象，例如，"input":{"ref":"myDataNodeId"} |
| lateAfterTimeout | 管道启动后经过的时间，在此时间内，对象必须完成。仅当计划类型未设置为 ondemand 时才会触发。 | 周期 |
| maxActiveInstances | 组件的并发活动实例的最大数量。重新运行不计入活动实例数中。 | 整数 |
| maximumRetries | 失败后的最大重试次数 | 整数 |
| onFail | 当前对象失败时要运行的操作。 | 引用对象，例如，"onFail":{"ref":"myActionId"} |
| onLateAction | 在尚未计划对象或对象仍未完成的情况下将触发的操作。 | 引用对象，例如，"onLateAction":{"ref":"myActionId"} |
| onSuccess | 当前对象成功时要运行的操作。 | 引用对象，例如，"onSuccess":{"ref":"myActionId"} |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| output | 输出数据源。 | 引用对象，例如，"output":{"ref":"myDataNodeId"} |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |
| pipelineLogUri | 用于上传管道日志的 S3 URI (例如“s3://BucketName/Key/”)。 | String |
| precondition | (可选) 定义先决条件。在满足所有先决条件之前，数据节点不会标记为“READY”。 | 引用对象，例如，"precondition":{"ref":"myPreconditionId"} |
| reportProgressTimeout | 远程工作对 reportProgress 的连续调用的超时时间。如果设置此字段，则未报告指定时段的进度的远程活动可能会被视为停滞且已重试。 | 周期 |
| retryDelay | 两次重试之间的超时时间。 | 周期 |
| scheduleType | 计划类型允许您指定应在间隔的结尾还是开头计划您管道定义中的对象。时间序列风格计划表示在每次间隔的结尾计划实例，而 Cron 风格计划表示应在每次间隔的开头计划实例。按需计划让您可以在每次激活时运行一次管道。这意味着，您不需要克隆或重新创建管道以再次运行它。如果您使用按需计划，则必须在默认对象中指定它，并且必须是在管道中为对象指定的唯一 scheduleType。要使用按需管道，您只需为每个后续运行调用 ActivatePipeline 操作。值包括：cron、ondemand 和 timeseries。 | 枚举 |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @activeInstances | 当前计划的有效实例对象的列表。 | 引用对象，例如，"activeInstances":{"ref":"myRunnableObjectId"} |
| @actualEndTime | 该对象的执行完成时间。 | DateTime |
| @actualStartTime | 该对象的执行开始时间。 | DateTime |
| cancellationReason | 该对象被取消时显示的 cancellationReason。 | String |
| @cascadeFailedOn | 对象在其上失败的依赖项链的描述。 | 引用对象，例如，"cascadeFailedOn":{"ref":"myRunnableObjectId"} |
| emrStepLog | 仅在尝试 EMR 活动时可用的 EMR 步骤日志 | String |
| errorId | 该对象失败时显示的 errorId。 | String |
| errorMessage | 该对象失败时显示的 errorMessage。 | String |
| errorStackTrace | 该对象失败时显示的错误堆栈跟踪。 | String |
| @finishedTime | 该对象完成其执行的时间。 | DateTime |
| hadoopJobLog | 在尝试基于 EMR 的活动时可用的 Hadoop 任务日志。 | String |
| @healthStatus | 对象的运行状况，反映进入终止状态的上个对象实例成功还是失败。 | String |
| @healthStatusFromInstanceId | 进入终止状态的上个实例对象的 ID。 | String |
| @healthStatusUpdatedTime | 上次更新运行状况的时间。 | DateTime |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| hostname | 已执行任务尝试的客户端的主机名。 | String |
| @lastDeactivatedTime | 上次停用该对象的时间。 | DateTime |
| @latestCompletedRunTime | 已完成执行的最新运行的时间。 | DateTime |
| @latestRunTime | 已计划执行的最新运行的时间。 | DateTime |
| @nextRunTime | 计划下次运行的时间。 | DateTime |
| reportProgressTime | 远程活动报告进度的最近时间。 | DateTime |
| @scheduledEndTime | 对象的计划结束时间。 | DateTime |
| @scheduledStartTime | 对象的计划开始时间。 | DateTime |
| @status | 该对象的状态。 | String |
| @version | 用来创建对象的管道版本。 | String |
| @waitingOn | 该对象在其上处于等待状态的依赖项列表的描述。 | 引用对象，例如，"waitingOn":{"ref":"myRunnableObjectId"} |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息 | String |
| @pipelineId | 该对象所属的管道的 ID | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象 | String |

### 另请参阅

- ShellCommandActivity
- EmrActivity
- 使用 AWS Data Pipeline 将 MySQL 数据导出到 Amazon S3。

## EmrActivity

运行 EMR 集群。

AWS Data Pipeline 对步骤使用的格式不同于 Amazon EMR 对步骤使用的格式；例如，AWS Data Pipeline 在 `EmrActivity` 步骤字段中的 JAR 名称后使用逗号分隔的参数。以下示例显示格式适合 Amazon EMR 的步骤，后跟其 AWS Data Pipeline 等效步骤：

```
s3://example-bucket/MyWork.jar arg1 arg2 arg3
```

```
"s3://example-bucket/MyWork.jar,arg1,arg2,arg3"
```

### 示例

以下是该对象类型的示例。该示例使用较旧版本的 Amazon EMR。验证该示例是否适合您使用的 Amazon EMR 集群版本。

该对象引用您将在同一管道定义文件中定义的其他三个对象。`MyEmrCluster` 为 `EmrCluster` 对象，`MyS3Input` 和 `MyS3Output` 为 `S3DataNode` 对象。

> **Note**
>
> 在此示例中，您可以将 `step` 字段替换为所需集群字符串，它可以是 Pig 脚本、Hadoop 流式集群、您自己的自定义 JAR (包括其参数) 等。

Hadoop 2.x (AMI 3.x)

```
{
  "id" : "MyEmrActivity",
  "type" : "EmrActivity",
  "runsOn" : { "ref" : "MyEmrCluster" },
  "preStepCommand" : "scp remoteFiles localFiles",
```

```
  "step" : ["s3://mybucket/myPath/myStep.jar,firstArg,secondArg,-files,s3://mybucket/
myPath/myFile.py,-input,s3://myinputbucket/path,-output,s3://myoutputbucket/path,-
mapper,myFile.py,-reducer,reducerName","s3://mybucket/myPath/myotherStep.jar,..."],
  "postStepCommand" : "scp localFiles remoteFiles",
  "input" : { "ref" : "MyS3Input" },
  "output" : { "ref" : "MyS3Output" }
}
```

**Note**

要通过一个步骤将参数传递给应用程序，您需要在脚本路径中指定区域，如以下示例所示：此外，您可能需要对传递的参数进行转义。例如，如果您使用 `script-runner.jar` 运行一个 Shell 脚本，并且需要将参数传递给该脚本，则必须对用于分隔参数的逗号进行转义。以下步骤槽介绍了如何执行此操作：

```
"step" : "s3://eu-west-1.elasticmapreduce/libs/script-runner/script-
runner.jar,s3://datapipeline/echo.sh,a\\\\,b\\\\,c"
```

此步骤使用 `script-runner.jar` 运行 `echo.sh` Shell 脚本，并将 a、b 和 c 作为单个参数传递给此脚本。由于将从生成的参数中删除第一个转义字符，因此，您可能需要重新转义。例如，如果您将 `File\.gz` 作为参数 (用 JSON 表示)，则可使用 `File\\\\.gz` 对其进行转义。但由于已丢弃第一个转义，因此，您必须使用 `File\\\\\\\\.gz` 。

## 语法

| 对象调用字段 | 描述 | 槽类型 |
| --- | --- | --- |
| 计划 | 该对象在计划间隔的执行中调用。指定对另一个对象的计划引用，以便设置该对象的依赖项执行顺序。您可以明确设置针对该对象的计划以满足该要求，例如，指定 `"schedule": {"ref": "DefaultSchedule"}` 。在大多数情况下，最好将计划引用放在默认管道对象上，以便所有对象继承该计划。或者，如果管道具有一个计划树 (计划位于主计划中)，您可以创建具有计划引用的父对象。有关示例可选计划配置的更多信息，请参阅 https://docs.aws. | 引用对象，例如，"schedule":{"ref":"myScheduleId"} |

| 对象调用字段 | 描述 | 槽类型 |
| --- | --- | --- |
| | amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html。 | |

| 所需的组 (下列选项之一是必需的) | 描述 | 槽类型 |
| --- | --- | --- |
| runsOn | 在其中运行该作业的 Amazon EMR 集群。 | 引用对象，例如，"runsOn":{"ref":"myEmrClusterId"} |
| workerGroup | 工作线程组。这可用于路由任务。如果您提供 `runsOn` 值并且存在 `workerGroup` ，则将忽略 `workerGroup` | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| attemptStatus | 来自远程活动的最近报告的状态。 | String |
| attemptTimeout | 远程工作完成的超时时间。如果设置此字段，则可能会重试未在设定的开始时间内完成的远程活动。 | 周期 |
| dependsOn | 指定与另一个可运行对象的依赖关系。 | 引用对象，例如，"dependsOn":{"ref":"myActivityId"} |
| failureAndRerunMode | 描述依赖项失败或重新运行时的使用者节点行为。 | 枚举 |
| input | 输入数据的位置。 | 引用对象，例如，"input":{"ref":"myDataNodeId"} |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| lateAfterTimeout | 管道启动后经过的时间，在此时间内，对象必须完成。仅当计划类型未设置为 ondemand 时才会触发。 | 周期 |
| maxActiveInstances | 组件的并发活动实例的最大数量。重新运行不计入活动实例数中。 | 整数 |
| maximumRetries | 失败后的最大重试次数。 | 整数 |
| onFail | 当前对象失败时要运行的操作。 | 引用对象，例如，"onFail":{"ref":"myActionId"} |
| onLateAction | 在尚未计划对象或对象仍未完成的情况下将触发的操作。 | 引用对象，例如，"onLateAction":{"ref":"myActionId"} |
| onSuccess | 当前对象成功时要运行的操作。 | 引用对象，例如，"onSuccess":{"ref":"myActionId"} |
| output | 输出数据的位置。 | 引用对象，例如，"output":{"ref":"myDataNodeId"} |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |
| pipelineLogUri | 用于上传管道日志的 Amazon S3 URI，例如，“s3://BucketName/Prefix/”。 | String |
| postStepCommand | 所有步骤完成之后运行的 Shell 脚本。要指定多个脚本 (最多 255 个)，请添加多个 postStepCommand 字段。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| precondition | (可选) 定义先决条件。在满足所有先决条件之前，数据节点不会标记为“READY”。 | 引用对象，例如，"precondition":{"ref":"myPreconditionId"} |
| preStepCommand | 在任意步骤运行之前运行的 Shell 脚本。要指定多个脚本 (最多 255 个)，请添加多个 `preStepCommand` 字段。 | String |
| reportProgressTimeout | 远程工作对 `reportProgress` 的连续调用的超时时间。如果设置此字段，则未报告指定时段的进度的远程活动可能会被视为停滞且已重试。 | 周期 |
| resizeClusterBeforeRunning | 在执行此活动前，重新调整集群的大小，以适应指定为输入或输出的 DynamoDB 表。<br><br>Note<br>如果您的 `EmrActivity` 使用 `DynamoDBDataNode` 作为输入或输出数据节点，而且您将 `resizeClusterBeforeRunning` 设置为 `TRUE`，AWS Data Pipeline 将开始使用 `m3.xlarge` 实例类型。这将使用 `m3.xlarge` 覆盖您的实例类型，从而可能会增加您的月度成本。 | 布尔值 |
| resizeClusterMaxInstances | 调整大小算法可以请求的最大实例数的限制。 | 整数 |
| retryDelay | 两次重试之间的超时时间。 | 周期 |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| scheduleType | 您可以通过计划类型指定应在间隔开头还是结尾计划管道定义中的对象。值包括：cron、ondemand 和 timeseries 。timeseries 计划表示在每个间隔结尾计划实例。cron 计划表示在每个间隔开头计划实例。ondemand 计划让您可以在每次激活时运行一次管道。您不需要克隆或重新创建管道以再次运行它。如果您使用 ondemand 计划，则必须在默认对象中指定它，并且该计划必须是在管道中为对象指定的唯一 scheduleType 。要使用 ondemand 管道，请为每个后续运行调用 ActivatePipeline 操作。 | 枚举 |
| step | 集群要运行的一个或多个步骤。要指定多个步骤 (最多 255 个)，请添加多个步骤字段。请在 JAR 名称后面使用以逗号分隔的参数，例如，“s3://example-bucket/MyWork.jar,arg1,arg2,arg3 ”。 | String |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @activeInstances | 当前计划的有效实例对象的列表。 | 引用对象，例如，"activeInstances":{"ref":"myRunnableObjectId"} |
| @actualEndTime | 该对象的执行完成时间。 | DateTime |
| @actualStartTime | 该对象的执行开始时间。 | DateTime |
| cancellationReason | 该对象被取消时显示的 cancellationReason。 | String |
| @cascadeFailedOn | 对象在其上失败的依赖项链的描述。 | 引用对象，例如，"cascadeFailedOn":{ |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| | | "ref":"myRunnableObjectId"} |
| emrStepLog | 仅在尝试 EMR 活动时可用的 Amazon EMR 步骤日志 | String |
| errorId | 该对象失败时显示的 `errorId`。 | String |
| errorMessage | 该对象失败时显示的 `errorMessage` 。 | String |
| errorStackTrace | 该对象失败时显示的错误堆栈跟踪。 | String |
| @finishedTime | 该对象完成其执行的时间。 | DateTime |
| hadoopJobLog | 在尝试基于 EMR 的活动时可用的 Hadoop 任务日志。 | String |
| @healthStatus | 对象的运行状况，反映进入终止状态的上个对象实例成功还是失败。 | String |
| @healthStatusFromInstanceId | 进入终止状态的上个实例对象的 ID。 | String |
| @healthStatusUpdatedTime | 上次更新运行状况的时间。 | DateTime |
| hostname | 已执行任务尝试的客户端的主机名。 | String |
| @lastDeactivatedTime | 上次停用该对象的时间。 | DateTime |
| @latestCompletedRunTime | 已完成执行的最新运行的时间。 | DateTime |
| @latestRunTime | 已计划执行的最新运行的时间。 | DateTime |
| @nextRunTime | 计划下次运行的时间。 | DateTime |
| reportProgressTime | 远程活动报告进度的最近时间。 | DateTime |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @scheduledEndTime | 对象的计划结束时间。 | DateTime |
| @scheduledStartTime | 对象的计划开始时间。 | DateTime |
| @status | 该对象的状态。 | String |
| @version | 用于创建对象的管道版本。 | String |
| @waitingOn | 该对象在其上处于等待状态的依赖项列表的描述。 | 引用对象，例如，"waitingOn":{"ref": "myRunnableObjectId"} |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息。 | String |
| @pipelineId | 该对象所属的管道的 ID。 | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象。 | String |

### 另请参阅

- ShellCommandActivity
- CopyActivity
- EmrCluster

## HadoopActivity

在集群上运行 MapReduce 作业。该集群可以是由 AWS Data Pipeline 或其他资源（如果您使用 TaskRunner）托管的 EMR 集群。在需要并行运行工作时，可使用 HadoopActivity。这使您能够在 Hadoop 1 中使用 YARN 框架的计划资源或 MapReduce 资源导航器。如果您想使用 Amazon EMR 步骤操作来按顺序运行工作，您仍可使用 EmrActivity。

## 示例

使用由 AWS Data Pipeline 管理的 EMR 集群的 HadoopActivity

以下 HadoopActivity 对象使用 EmrCluster 资源来运行程序：

```
 {
   "name": "MyHadoopActivity",
   "schedule": {"ref": "ResourcePeriod"},
   "runsOn": {"ref": “MyEmrCluster”},
   "type": "HadoopActivity",
   "preActivityTaskConfig":{"ref":"preTaskScriptConfig”},
   "jarUri": "/home/hadoop/contrib/streaming/hadoop-streaming.jar",
   "argument": [
     "-files",
     “s3://elasticmapreduce/samples/wordcount/wordSplitter.py“,
     "-mapper",
     "wordSplitter.py",
     "-reducer",
     "aggregate",
     "-input",
     "s3://elasticmapreduce/samples/wordcount/input/",
     "-output",
     “s3://test-bucket/MyHadoopActivity/#{@pipelineId}/
#{format(@scheduledStartTime,'YYYY-MM-dd')}"
   ],
   "maximumRetries": "0",
   "postActivityTaskConfig":{"ref":"postTaskScriptConfig”},
   "hadoopQueue" : “high”
 }
```

以下是相应的 *MyEmrCluster*，它将配置 FairScheduler 并针对基于 Hadoop 2 的 AMI 在 YARN 中排队：

```
{
  "id" : "MyEmrCluster",
  "type" : "EmrCluster",
   "hadoopSchedulerType" : "PARALLEL_FAIR_SCHEDULING",
  “amiVersion” : “3.7.0”,
  "bootstrapAction" : ["s3://Region.elasticmapreduce/bootstrap-
actions/configure-hadoop,-z,yarn.scheduler.capacity.root.queues=low
\,high\,default,-z,yarn.scheduler.capacity.root.high.capacity=50,-
```

```
z,yarn.scheduler.capacity.root.low.capacity=10,-
z,yarn.scheduler.capacity.root.default.capacity=30”]
}
```

这是您用于在 Hadoop 1 中配置 FairScheduler 的 EmrCluster：

```
{
      "id": "MyEmrCluster",
      "type": "EmrCluster",
      "hadoopSchedulerType": "PARALLEL_FAIR_SCHEDULING",
      "amiVersion": "2.4.8",
      "bootstrapAction": "s3://Region.elasticmapreduce/bootstrap-
actions/configure-hadoop,-m,mapred.queue.names=low\\\\,high\\\\,default,-
m,mapred.fairscheduler.poolnameproperty=mapred.job.queue.name"
          }
```

以下 EmrCluster 为基于 Hadoop 的 AMI 配置 CapacityScheduler：

```
{
      "id": "MyEmrCluster",
      "type": "EmrCluster",
      "hadoopSchedulerType": "PARALLEL_CAPACITY_SCHEDULING",
      "amiVersion": "3.7.0",
      "bootstrapAction": "s3://Region.elasticmapreduce/bootstrap-
actions/configure-hadoop,-z,yarn.scheduler.capacity.root.queues=low
\\\\,high,-z,yarn.scheduler.capacity.root.high.capacity=40,-
z,yarn.scheduler.capacity.root.low.capacity=60"
    }
```

使用现有 EMR 集群的 HadoopActivity

在此示例中，您将使用工作线程组和 TaskRunner 在现有 EMR 集群上运行程序。以下管道定义使用 HadoopActivity 执行以下操作：

- 仅在 *myWorkerGroup* 资源上运行 MapReduce 程序。有关工作线程组的更多信息，请参阅使用任务运行程序在现有资源上执行工作。
- 运行 preActivityTaskConfig 和 postActivityTaskConfig

```
{
  "objects": [
    {
```

```
      "argument": [
        "-files",
        "s3://elasticmapreduce/samples/wordcount/wordSplitter.py",
        "-mapper",
        "wordSplitter.py",
        "-reducer",
        "aggregate",
        "-input",
        "s3://elasticmapreduce/samples/wordcount/input/",
        "-output",
        "s3://test-bucket/MyHadoopActivity/#{@pipelineId}/
#{format(@scheduledStartTime,'YYYY-MM-dd')}"
      ],
      "id": "MyHadoopActivity",
      "jarUri": "/home/hadoop/contrib/streaming/hadoop-streaming.jar",
      "name": "MyHadoopActivity",
      "type": "HadoopActivity"
    },
    {
      "id": "SchedulePeriod",
      "startDateTime": "start_datetime",
      "name": "SchedulePeriod",
      "period": "1 day",
      "type": "Schedule",
      "endDateTime": "end_datetime"
    },
    {
      "id": "ShellScriptConfig",
      "scriptUri": "s3://test-bucket/scripts/preTaskScript.sh",
      "name": "preTaskScriptConfig",
      "scriptArgument": [
        "test",
        "argument"
      ],
      "type": "ShellScriptConfig"
    },
    {
      "id": "ShellScriptConfig",
      "scriptUri": "s3://test-bucket/scripts/postTaskScript.sh",
      "name": "postTaskScriptConfig",
      "scriptArgument": [
        "test",
        "argument"
      ],
```

```
      "type": "ShellScriptConfig"
    },
    {
      "id": "Default",
      "scheduleType": "cron",
      "schedule": {
        "ref": "SchedulePeriod"
      },
      "name": "Default",
      "pipelineLogUri": "s3://test-bucket/logs/2015-05-22T18:02:00.343Z642f3fe415",
      "maximumRetries": "0",
      "workerGroup": "myWorkerGroup",
      "preActivityTaskConfig": {
        "ref": "preTaskScriptConfig"
      },
      "postActivityTaskConfig": {
        "ref": "postTaskScriptConfig"
      }
    }
  ]
}
```

## 语法

| 必填字段 | 描述 | 槽类型 |
| --- | --- | --- |
| jarUri | 与 HadoopActivity 一起运行的 Amazon S3 中 JAR 或集群本地文件系统的位置。 | String |

| 对象调用字段 | 描述 | 槽类型 |
| --- | --- | --- |
| 计划 | 该对象在计划间隔的执行中调用。用户必须指定对另一个对象的计划引用，以便设置该对象的依赖项执行顺序。用户可在对象上明确设置计划来满足此要求，例如，指定 "schedule": {"ref": "DefaultSchedule"}。在大多数情况下，最好将计划引用放在默认管道对象上，以便所有对象继承该计划。或者，如果管道有一个计 | 引用对象，例如，"schedule":{"ref":"myScheduleId"} |

| 对象调用字段 | 描述 | 槽类型 |
| --- | --- | --- |
| | 划树 (计划位于主计划中)，用户可以创建具有计划引用的父对象。有关示例可选计划配置的更多信息，请参阅 https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html。 | |

| 所需的组 (下列选项之一是必需的) | 描述 | 槽类型 |
| --- | --- | --- |
| runsOn | 运行此作业的 EMR 集群。 | 引用对象，例如，"runsOn":{"ref":"myEmrClusterId"} |
| workerGroup | 工作线程组。这可用于路由任务。如果您提供 runsOn 值并且存在 workerGroup，则将忽略 workerGroup。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| argument | 要传递给 JAR 的参数。 | String |
| attemptStatus | 来自远程活动的最近报告的状态。 | String |
| attemptTimeout | 远程工作完成的超时时间。如果设置此字段，则可能会重试未在设定的开始时间内完成的远程活动。 | 周期 |
| dependsOn | 指定与另一个可运行对象的依赖关系。 | 引用对象，例如，"dependsOn":{"ref":"myActivityId"} |
| failureAndRerunMode | 描述依赖项失败或重新运行时的使用者节点行为。 | 枚举 |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| hadoopQueue | 将在其上提交活动的 Hadoop 计划程序队列名。 | String |
| input | 输入数据的位置。 | 引用对象，例如，"input":{"ref":"myDataNodeId"} |
| lateAfterTimeout | 管道启动后经过的时间，在此时间内，对象必须完成。仅当计划类型未设置为 ondemand 时才会触发。 | 周期 |
| mainClass | 您与 HadoopActivity 一起执行的 JAR 的主类。 | String |
| maxActiveInstances | 组件的并发活动实例的最大数量。重新运行不计入活动实例数中。 | 整数 |
| maximumRetries | 失败后的最大重试次数 | 整数 |
| onFail | 当前对象失败时要运行的操作。 | 引用对象，例如，"onFail":{"ref":"myActionId"} |
| onLateAction | 在尚未计划对象或对象仍未完成的情况下将触发的操作。 | 引用对象，例如，"onLateAction":{"ref":"myActionId"} |
| onSuccess | 当前对象成功时要运行的操作。 | 引用对象，例如，"onSuccess":{"ref":"myActionId"} |
| output | 输出数据的位置。 | 引用对象，例如，"output":{"ref":"myDataNodeId"} |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| pipelineLogUri | 用于上传管道日志的 S3 URI (例如“s3://BucketName/Key/”)。 | String |
| postActivityTaskConfig | 要运行的活动后配置脚本。这由 Amazon S3 中 Shell 脚本的 URI 和参数列表组成。 | 引用对象，例如，"postActivityTaskConfig":{"ref":"myShellScriptConfigId"} |
| preActivityTaskConfig | 要运行的活动前配置脚本。这由 Amazon S3 中 Shell 脚本的 URI 和参数列表组成。 | 引用对象，例如，"preActivityTaskConfig":{"ref":"myShellScriptConfigId"} |
| precondition | (可选) 定义先决条件。在满足所有先决条件之前，数据节点不会标记为“READY”。 | 引用对象，例如，"precondition":{"ref":"myPreconditionId"} |
| reportProgressTimeout | 远程工作对 reportProgress 的连续调用的超时时间。如果设置此字段，则未报告指定时段的进度的远程活动可能会被视为停滞且已重试。 | 周期 |
| retryDelay | 两次重试之间的超时时间。 | 周期 |
| scheduleType | 计划类型允许您指定应在间隔的结尾还是开头计划您管道定义中的对象。时间序列风格计划表示在每次间隔的结尾计划实例，而 Cron 风格计划表示应在每次间隔的开头计划实例。按需计划让您可以在每次激活时运行一次管道。这意味着，您不需要克隆或重新创建管道以再次运行它。如果您使用按需计划，则必须在默认对象中指定它，并且必须是在管道中为对象指定的唯一 scheduleType。要使用按需管道，您只需为每个后续运行调用 ActivatePipeline 操作。值包括：cron、ondemand 和 timeseries。 | 枚举 |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @activeInstances | 当前计划的有效实例对象的列表。 | 引用对象，例如，"activeInstances":{"ref":"myRunnableObjectId"} |
| @actualEndTime | 该对象的执行完成时间。 | DateTime |
| @actualStartTime | 该对象的执行开始时间。 | DateTime |
| cancellationReason | 该对象被取消时显示的 cancellationReason。 | String |
| @cascadeFailedOn | 对象在其上失败的依赖项链的描述。 | 引用对象，例如，"cascadeFailedOn":{"ref":"myRunnableObjectId"} |
| emrStepLog | 仅在尝试 EMR 活动时可用的 EMR 步骤日志 | String |
| errorId | 该对象失败时显示的 errorId。 | String |
| errorMessage | 该对象失败时显示的 errorMessage。 | String |
| errorStackTrace | 该对象失败时显示的错误堆栈跟踪。 | String |
| @finishedTime | 该对象完成其执行的时间。 | DateTime |
| hadoopJobLog | 在尝试基于 EMR 的活动时可用的 Hadoop 任务日志。 | String |
| @healthStatus | 对象的运行状况，反映进入终止状态的上个对象实例成功还是失败。 | String |
| @healthStatusFromInstanceId | 进入终止状态的上个实例对象的 ID。 | String |
| @healthStatusUpdatedTime | 上次更新运行状况的时间。 | DateTime |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| hostname | 已执行任务尝试的客户端的主机名。 | String |
| @lastDeactivatedTime | 上次停用该对象的时间。 | DateTime |
| @latestCompletedRunTime | 已完成执行的最新运行的时间。 | DateTime |
| @latestRunTime | 已计划执行的最新运行的时间。 | DateTime |
| @nextRunTime | 计划下次运行的时间。 | DateTime |
| reportProgressTime | 远程活动报告进度的最近时间。 | DateTime |
| @scheduledEndTime | 对象的计划结束时间。 | DateTime |
| @scheduledStartTime | 对象的计划开始时间。 | DateTime |
| @status | 该对象的状态。 | String |
| @version | 用来创建对象的管道版本。 | String |
| @waitingOn | 该对象在其上处于等待状态的依赖项列表的描述。 | 引用对象，例如，"waitingOn":{"ref":"myRunnableObjectId"} |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息。 | String |
| @pipelineId | 该对象所属的管道的 ID。 | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象。 | String |

### 另请参阅

- ShellCommandActivity
- CopyActivity
- EmrCluster

## HiveActivity

在 EMR 集群上运行 Hive 查询。HiveActivity 使您能够更轻松地设置 Amazon EMR 活动，并且将自动基于来自 Amazon S3 或 Amazon RDS 的输入数据创建 Hive 表。您只需指定要在源数据上运行的 HiveQL。AWS Data Pipeline 将自动使用 ${input1}、${input2} 等基于 HiveActivity 对象中的输入字段创建 Hive 表。

对于 Amazon S3 输入，dataFormat 字段用于创建 Hive 列名。

对于 MySQL (Amazon RDS) 输入，SQL 查询的列名用于创建 Hive 列名。

> **Note**
>
> 此活动使用 Hive CSV Serde。

### 示例

以下是该对象类型的示例。该对象引用您在同一管道定义文件中定义的三个其他对象。MySchedule 为 Schedule 对象，MyS3Input 和 MyS3Output 为数据节点对象。

```
{
  "name" : "ProcessLogData",
  "id" : "MyHiveActivity",
  "type" : "HiveActivity",
  "schedule" : { "ref": "MySchedule" },
  "hiveScript" : "INSERT OVERWRITE TABLE ${output1} select
 host,user,time,request,status,size from ${input1};",
  "input" : { "ref": "MyS3Input" },
  "output" : { "ref": "MyS3Output" },
  "runsOn" : { "ref": "MyEmrCluster" }
}
```

## 语法

| 对象调用字段 | 描述 | 槽类型 |
| --- | --- | --- |
| 计划 | 该对象在计划间隔的执行中调用。指定对另一个对象的计划引用，以便设置该对象的依赖项执行顺序。您可以明确设置针对该对象的计划以满足该要求，例如，指定 "schedule": {"ref": "DefaultSchedule"}。在大多数情况下，最好将计划引用放在默认管道对象上，以便所有对象继承该计划。或者，如果管道具有一个计划树 (计划位于主计划中)，您可以创建具有计划引用的父对象。有关示例可选计划配置的更多信息，请参阅 https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html。 | 引用对象，例如，"schedule":{"ref":"myScheduleId"} |

| 所需的组 (下列选项之一是必需的) | 描述 | 槽类型 |
| --- | --- | --- |
| hiveScript | 要运行的 Hive 脚本。 | String |
| scriptUri | 要运行 Hive 脚本的位置 (例如，s3://scriptLocation)。 | String |

| 所需的组 | 描述 | 槽类型 |
| --- | --- | --- |
| runsOn | 在其中运行该 `HiveActivity` 的 EMR 集群。 | 引用对象，例如，"runsOn":{"ref":"myEmrClusterId"} |
| workerGroup | 工作线程组。这可用于路由任务。如果您提供 `runsOn` 值并且存在 `workerGroup`，则将忽略 `workerGroup` | String |

| 所需的组 | 描述 | 槽类型 |
| --- | --- | --- |
| input | 输入数据源。 | 引用对象，例如，"input":{"ref":"myDataNodeId"} |
| output | 输出数据源。 | 引用对象，例如，"output":{"ref":"myDataNodeId"} |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| attemptStatus | 来自远程活动的最近报告的状态。 | String |
| attemptTimeout | 远程工作完成的超时时间。如果设置此字段，则可能会重试未在设定的开始时间内完成的远程活动。 | 周期 |
| dependsOn | 指定与另一个可运行对象的依赖关系。 | 引用对象，例如，"dependsOn":{"ref":"myActivityId"} |
| failureAndRerunMode | 描述依赖项失败或重新运行时的使用者节点行为。 | 枚举 |
| hadoopQueue | 将在其上提交作业的 Hadoop 计划程序队列名。 | String |
| lateAfterTimeout | 管道启动后经过的时间，在此时间内，对象必须完成。仅当计划类型未设置为 ondemand 时才会触发。 | 周期 |
| maxActiveInstances | 组件的并发活动实例的最大数量。重新运行不计入活动实例数中。 | 整数 |
| maximumRetries | 失败后的最大重试次数。 | 整数 |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| onFail | 当前对象失败时要运行的操作。 | 引用对象，例如，"onFail":{"ref":"myActionId"} |
| onLateAction | 在尚未计划对象或对象仍未完成的情况下将触发的操作。 | 引用对象，例如，"onLateAction":{"ref":"myActionId"} |
| onSuccess | 当前对象成功时要运行的操作。 | 引用对象，例如，"onSuccess":{"ref":"myActionId"} |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |
| pipelineLogUri | 用于上传管道日志的 S3 URI (例如“s3://BucketName/Key/”)。 | String |
| postActivityTaskConfig | 要运行的活动后配置脚本。这由 Amazon S3 中 Shell 脚本的 URI 和参数列表组成。 | 引用对象，例如，"postActivityTaskConfig":{"ref":"myShellScriptConfigId"} |
| preActivityTaskConfig | 要运行的活动前配置脚本。这由 Amazon S3 中 Shell 脚本的 URI 和参数列表组成。 | 引用对象，例如，"preActivityTaskConfig":{"ref":"myShellScriptConfigId"} |
| precondition | (可选) 定义先决条件。在满足所有先决条件之前，数据节点不会标记为“READY”。 | 引用对象，例如，"precondition":{"ref":"myPreconditionId"} |
| reportProgressTimeout | 远程工作对 `reportProgress` 的连续调用的超时时间。如果设置此字段，则未报告指定时段的进度的远程活动可能会被视为停滞且已重试。 | 周期 |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| resizeClusterBeforeRunning | 在执行此活动前，重新调整集群的大小，以适应指定为输入或输出的 DynamoDB 数据节点。<br><br>**Note**<br>如果您的活动使用 DynamoDBDataNode 作为输入或输出数据节点，而且您将 resizeClusterBeforeRunning 设置为 TRUE，AWS Data Pipeline 将开始使用 m3.xlarge 实例类型。这将使用 m3.xlarge 覆盖您的实例类型，从而可能会增加您的月度成本。 | 布尔值 |
| resizeClusterMaxInstances | 调整大小算法可以请求的最大实例数的限制。 | 整数 |
| retryDelay | 两次重试之间的超时时间。 | 周期 |
| scheduleType | 计划类型允许您指定应在间隔的结尾还是开头计划您管道定义中的对象。时间序列风格计划表示在每次间隔的结尾计划实例，而 Cron 风格计划表示应在每次间隔的开头计划实例。按需计划让您可以在每次激活时运行一次管道。这意味着，您不需要克隆或重新创建管道以再次运行它。如果您使用按需计划，则必须在默认对象中指定它，并且必须是在管道中为对象指定的唯一 scheduleType。要使用按需管道，您只需为每个后续运行调用 ActivatePipeline 操作。值包括：cron、ondemand 和 timeseries。 | 枚举 |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| scriptVariable | 指定在运行脚本时要传递给 Hive 的 Amazon EMR 的脚本变量。例如，以下示例脚本变量将 SAMPLE 和 FILTER_DATE 变量传递到 Hive：SAMPLE=s3://elasticmapreduce/samples/hive-ads 和 FILTER_DATE=#{format(@scheduledStartTime,'YYYY-MM-dd')}% 。此字段接受多个值，可与 script 和 scriptUri 字段结合使用。此外，scriptVariable 将起作用，不管 stage 设置为 true 还是 false。在使用 AWS Data Pipeline 表达式和函数将动态值发送到 Hive 时，此字段特别有用。 | String |
| stage | 确定是在运行脚本之前还是之后启用暂存。不可对 Hive 11 使用，因此，请使用 Amazon EMR AMI 版本 3.2.0 或更高版本。 | 布尔值 |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @activeInstances | 当前计划的有效实例对象的列表。 | 引用对象，例如，"activeInstances":{"ref":"myRunnableObjectId"} |
| @actualEndTime | 该对象的执行完成时间。 | DateTime |
| @actualStartTime | 该对象的执行开始时间。 | DateTime |
| cancellationReason | 该对象被取消时显示的 cancellationReason。 | String |
| @cascadeFailedOn | 对象在其上失败的依赖项链的描述。 | 引用对象，例如，"cascadeFailedOn":{"ref":"myRunnableObjectId"} |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| emrStepLog | 仅在尝试 EMR 活动时可用的 Amazon EMR 步骤日志。 | String |
| errorId | 该对象失败时显示的 errorId。 | String |
| errorMessage | 该对象失败时显示的 errorMessage。 | String |
| errorStackTrace | 该对象失败时显示的错误堆栈跟踪。 | String |
| @finishedTime | 该对象完成其执行的时间。 | DateTime |
| hadoopJobLog | 在尝试基于 EMR 的活动时可用的 Hadoop 任务日志。 | String |
| @healthStatus | 对象的运行状况，反映进入终止状态的上个对象实例成功还是失败。 | String |
| @healthStatusFromInstanceId | 进入终止状态的上个实例对象的 ID。 | String |
| @healthStatusUpdatedTime | 上次更新运行状况的时间。 | DateTime |
| hostname | 已执行任务尝试的客户端的主机名。 | String |
| @lastDeactivatedTime | 上次停用该对象的时间。 | DateTime |
| @latestCompletedRunTime | 已完成执行的最新运行的时间。 | DateTime |
| @latestRunTime | 已计划执行的最新运行的时间。 | DateTime |
| @nextRunTime | 计划下次运行的时间。 | DateTime |
| reportProgressTime | 远程活动报告进度的最近时间。 | DateTime |
| @scheduledEndTime | 对象的计划结束时间。 | DateTime |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @scheduledStartTime | 对象的计划开始时间。 | DateTime |
| @status | 该对象的状态。 | String |
| @version | 用来创建对象的管道版本。 | String |
| @waitingOn | 该对象在其上处于等待状态的依赖项列表的描述。 | 引用对象，例如，"waitingOn":{"ref": "myRunnableObjectId"} |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息。 | String |
| @pipelineId | 该对象所属的管道的 ID。 | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象。 | String |

### 另请参阅

- ShellCommandActivity
- EmrActivity

## HiveCopyActivity

在 EMR 集群上运行 Hive 查询。HiveCopyActivity 可让您更轻松地在 DynamoDB 表之间复制数据。HiveCopyActivity 接受 HiveQL 语句以在列和行级别筛选来自 DynamoDB 的输入数据。

### 示例

以下示例说明如何使用 HiveCopyActivity 和 DynamoDBExportDataFormat 将数据从一个 DynamoDBDataNode 复制到另一个 DynamoDBDataNode，并基于时间戳筛选数据。

```
{
  "objects": [
    {
      "id" : "DataFormat.1",
      "name" : "DataFormat.1",
      "type" : "DynamoDBExportDataFormat",
      "column" : "timeStamp BIGINT"
    },
    {
      "id" : "DataFormat.2",
      "name" : "DataFormat.2",
      "type" : "DynamoDBExportDataFormat"
    },
    {
      "id" : "DynamoDBDataNode.1",
      "name" : "DynamoDBDataNode.1",
      "type" : "DynamoDBDataNode",
      "tableName" : "item_mapped_table_restore_temp",
      "schedule" : { "ref" : "ResourcePeriod" },
      "dataFormat" : { "ref" : "DataFormat.1" }
    },
    {
      "id" : "DynamoDBDataNode.2",
      "name" : "DynamoDBDataNode.2",
      "type" : "DynamoDBDataNode",
      "tableName" : "restore_table",
      "region" : "us_west_1",
      "schedule" : { "ref" : "ResourcePeriod" },
      "dataFormat" : { "ref" : "DataFormat.2" }
    },
    {
      "id" : "EmrCluster.1",
      "name" : "EmrCluster.1",
      "type" : "EmrCluster",
      "schedule" : { "ref" : "ResourcePeriod" },
      "masterInstanceType" : "m1.xlarge",
      "coreInstanceCount" : "4"
    },
    {
      "id" : "HiveTransform.1",
      "name" : "Hive Copy Transform.1",
      "type" : "HiveCopyActivity",
      "input" : { "ref" : "DynamoDBDataNode.1" },
```

```
      "output" : { "ref" : "DynamoDBDataNode.2" },
      "schedule" :{ "ref" : "ResourcePeriod" },
      "runsOn" : { "ref" : "EmrCluster.1" },
      "filterSql" : "`timeStamp` > unix_timestamp(\"#{@scheduledStartTime}\", \"yyyy-
MM-dd'T'HH:mm:ss\")"
    },
    {
      "id" : "ResourcePeriod",
      "name" : "ResourcePeriod",
      "type" : "Schedule",
      "period" : "1 Hour",
      "startDateTime" : "2013-06-04T00:00:00",
      "endDateTime" : "2013-06-04T01:00:00"
    }
  ]
}
```

## 语法

| 对象调用字段 | 描述 | 槽类型 |
| --- | --- | --- |
| 计划 | 该对象在计划间隔的执行中调用。用户必须指定对另一个对象的计划引用，以便设置该对象的依赖项执行顺序。用户可在对象上明确设置计划来满足此要求，例如，指定 "schedule": {"ref": "DefaultSchedule"}。在大多数情况下，最好将计划引用放在默认管道对象上，以便所有对象继承该计划。或者，如果管道有一个计划树 (计划位于主计划中)，用户可以创建具有计划引用的父对象。有关示例可选计划配置的更多信息，请参阅 https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html。 | 引用对象，例如，"schedule":{"ref":"myScheduleId"} |

| 所需的组 (下列选项之一是必需的) | 描述 | 槽类型 |
| --- | --- | --- |
| runsOn | 指定在其上运行的集群。 | 引用对象，例如，"runsOn":{"ref":"myResourceId"} |
| workerGroup | 工作线程组。这可用于路由任务。如果您提供 `runsOn` 值并且存在 `workerGroup` ，则将忽略 `workerGroup` | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| attemptStatus | 来自远程活动的最近报告的状态。 | String |
| attemptTimeout | 远程工作完成的超时时间。如果设置此字段，则可能会重试未在设定的开始时间内完成的远程活动。 | 周期 |
| dependsOn | 指定与另一个可运行对象的依赖关系。 | 引用对象，例如，"dependsOn":{"ref":"myActivityId"} |
| failureAndRerunMode | 描述依赖项失败或重新运行时的使用者节点行为。 | 枚举 |
| filterSql | Hive SQL 语句片段，用于筛选要复制的 DynamoDB 或 Amazon S3 数据的子集。筛选器应仅包含谓词，不能以 `WHERE` 子句开头，因为 AWS Data Pipeline 会自动添加它。 | String |
| input | 输入数据源。这必须是 `S3DataNode` 或 `DynamoDBDataNode` 。如果您使用 `DynamoDBNode` ，请指定 `DynamoDBExportDataFormat` 。 | 引用对象，例如，"input":{"ref":"myDataNodeId"} |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| lateAfterTimeout | 管道启动后经过的时间，在此时间内，对象必须完成。仅当计划类型未设置为 ondemand 时才会触发。 | 周期 |
| maxActiveInstances | 组件的并发活动实例的最大数量。重新运行不计入活动实例数中。 | 整数 |
| maximumRetries | 失败后的最大重试次数。 | 整数 |
| onFail | 当前对象失败时要运行的操作。 | 引用对象，例如，"onFail":{"ref":"myActionId"} |
| onLateAction | 在尚未计划对象或对象仍未完成的情况下将触发的操作。 | 引用对象，例如，"onLateAction":{"ref":"myActionId"} |
| onSuccess | 当前对象成功时要运行的操作。 | 引用对象，例如，"onSuccess":{"ref":"myActionId"} |
| output | 输出数据源。如果输入是 S3DataNode ，则这必须是 DynamoDBDataNode 。否则，这可以是 S3DataNode 或 DynamoDBDataNode 。如果您使用 DynamoDBNode ，请指定 DynamoDBExportDataFormat 。 | 引用对象，例如，"output":{"ref":"myDataNodeId"} |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |
| pipelineLogUri | 用于上传管道日志的 Amazon S3 URI，例如 's3://BucketName/Key/' 。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| postActivityTaskConfig | 要运行的活动后配置脚本。这由 Amazon S3 中 Shell 脚本的 URI 和参数列表组成。 | 引用对象，例如，"postActivityTaskConfig":{"ref":"myShellScriptConfigId"} |
| preActivityTaskConfig | 要运行的活动前配置脚本。这由 Amazon S3 中 Shell 脚本的 URI 和参数列表组成。 | 引用对象，例如，"preActivityTaskConfig":{"ref":"myShellScriptConfigId"} |
| precondition | (可选) 定义先决条件。在满足所有先决条件之前，数据节点不会标记为“READY”。 | 引用对象，例如，"precondition":{"ref":"myPreconditionId"} |
| reportProgressTimeout | 远程工作对 `reportProgress` 的连续调用的超时时间。如果设置此字段，则未报告指定时段的进度的远程活动可能会被视为停滞且已重试。 | 周期 |
| resizeClusterBeforeRunning | 在执行此活动前，重新调整集群的大小，以适应指定为输入或输出的 DynamoDB 数据节点。<br><br>**Note**<br>如果您的活动使用 `DynamoDBDataNode` 作为输入或输出数据节点，而且您将 `resizeClusterBeforeRunning` 设置为 `TRUE`，AWS Data Pipeline 将开始使用 `m3.xlarge` 实例类型。这将使用 `m3.xlarge` 覆盖您的实例类型，从而可能会增加您的月度成本。 | 布尔值 |
| resizeClusterMaxInstances | 调整大小算法可以请求的最大实例数的限制 | 整数 |
| retryDelay | 两次重试之间的超时时间。 | 周期 |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| scheduleType | 计划类型允许您指定应在间隔的结尾还是开头计划您管道定义中的对象。时间序列风格计划表示在每次间隔的结尾计划实例，而 Cron 风格计划表示应在每次间隔的开头计划实例。按需计划让您可以在每次激活时运行一次管道。这意味着，您不需要克隆或重新创建管道以再次运行它。如果您使用按需计划，则必须在默认对象中指定它，并且必须是在管道中为对象指定的唯一 scheduleType。要使用按需管道，您只需为每个后续运行调用 ActivatePipeline 操作。值包括：cron、ondemand 和 timeseries。 | 枚举 |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @activeInstances | 当前计划的有效实例对象的列表。 | 引用对象，例如，"activeInstances":{"ref":"myRunnableObjectId"} |
| @actualEndTime | 该对象的执行完成时间。 | DateTime |
| @actualStartTime | 该对象的执行开始时间。 | DateTime |
| cancellationReason | 该对象被取消时显示的 cancellationReason。 | String |
| @cascadeFailedOn | 对象在其上失败的依赖项链的描述。 | 引用对象，例如，"cascadeFailedOn":{"ref":"myRunnableObjectId"} |
| emrStepLog | 仅在尝试 EMR 活动时可用的 Amazon EMR 步骤日志。 | String |
| errorId | 该对象失败时显示的 errorId。 | String |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| errorMessage | 该对象失败时显示的 errorMessage。 | String |
| errorStackTrace | 该对象失败时显示的错误堆栈跟踪。 | String |
| @finishedTime | 该对象完成其执行的时间。 | DateTime |
| hadoopJobLog | 在尝试基于 EMR 的活动时可用的 Hadoop 任务日志。 | String |
| @healthStatus | 对象的运行状况，反映进入终止状态的上个对象实例成功还是失败。 | String |
| @healthStatusFromInstanceId | 进入终止状态的上个实例对象的 ID。 | String |
| @healthStatusUpdatedTime | 上次更新运行状况的时间。 | DateTime |
| hostname | 已执行任务尝试的客户端的主机名。 | String |
| @lastDeactivatedTime | 上次停用该对象的时间。 | DateTime |
| @latestCompletedRunTime | 已完成执行的最新运行的时间。 | DateTime |
| @latestRunTime | 已计划执行的最新运行的时间。 | DateTime |
| @nextRunTime | 计划下次运行的时间。 | DateTime |
| reportProgressTime | 远程活动报告进度的最近时间。 | DateTime |
| @scheduledEndTime | 对象的计划结束时间。 | DateTime |
| @scheduledStartTime | 对象的计划开始时间。 | DateTime |
| @status | 该对象的状态。 | String |
| @version | 用来创建对象的管道版本。 | String |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @waitingOn | 该对象在其上处于等待状态的依赖项列表的描述。 | 引用对象，例如，"waitingOn":{"ref": "myRunnableObjectId"} |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息。 | String |
| @pipelineId | 该对象所属的管道的 ID。 | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象。 | String |

### 另请参阅

- ShellCommandActivity
- EmrActivity

## PigActivity

PigActivity 提供对 AWS Data Pipeline 中的 Pig 脚本的本机支持，而不需要使用 `ShellCommandActivity` 或 `EmrActivity`。此外，PigActivity 支持数据暂存。在将 stage 字段设置为 true 时，AWS Data Pipeline 会在 Pig 中将输入数据暂存为架构，而无需用户编写其他代码。

### 示例

以下示例管道说明如何使用 `PigActivity`。该示例管道执行以下步骤：

- MyPigActivity1 从 Amazon S3 加载数据并运行一个 Pig 脚本，此脚本将选择几个数据列并将其上传到 Amazon S3。
- MyPigActivity2 加载第一个输出，选择几个数据列和 3 个数据行，然后将其作为另一个输出上传到 Amazon S3。

- MyPigActivity3 加载第二个输出数据，并将两个数据行和仅名为“fifth”的列插入 Amazon RDS。
- MyPigActivity4 加载 Amazon RDS 数据，选择第一个数据行并将其上传到 Amazon S3。

```
{
  "objects": [
    {
      "id": "MyInputData1",
      "schedule": {
        "ref": "MyEmrResourcePeriod"
      },
      "directoryPath": "s3://example-bucket/pigTestInput",
      "name": "MyInputData1",
      "dataFormat": {
        "ref": "MyInputDataType1"
      },
      "type": "S3DataNode"
    },
    {
      "id": "MyPigActivity4",
      "scheduleType": "CRON",
      "schedule": {
        "ref": "MyEmrResourcePeriod"
      },
      "input": {
        "ref": "MyOutputData3"
      },
      "pipelineLogUri": "s3://example-bucket/path/",
      "name": "MyPigActivity4",
      "runsOn": {
        "ref": "MyEmrResource"
      },
      "type": "PigActivity",
      "dependsOn": {
        "ref": "MyPigActivity3"
      },
      "output": {
        "ref": "MyOutputData4"
      },
      "script": "B = LIMIT ${input1} 1; ${output1} = FOREACH B GENERATE one;",
      "stage": "true"
    },
    {
```

```
      "id": "MyPigActivity3",
      "scheduleType": "CRON",
      "schedule": {
        "ref": "MyEmrResourcePeriod"
      },
      "input": {
        "ref": "MyOutputData2"
      },
      "pipelineLogUri": "s3://example-bucket/path",
      "name": "MyPigActivity3",
      "runsOn": {
        "ref": "MyEmrResource"
      },
      "script": "B = LIMIT ${input1} 2; ${output1} = FOREACH B GENERATE Fifth;",
      "type": "PigActivity",
      "dependsOn": {
        "ref": "MyPigActivity2"
      },
      "output": {
        "ref": "MyOutputData3"
      },
      "stage": "true"
    },
    {
      "id": "MyOutputData2",
      "schedule": {
        "ref": "MyEmrResourcePeriod"
      },
      "name": "MyOutputData2",
      "directoryPath": "s3://example-bucket/PigActivityOutput2",
      "dataFormat": {
        "ref": "MyOutputDataType2"
      },
      "type": "S3DataNode"
    },
    {
      "id": "MyOutputData1",
      "schedule": {
        "ref": "MyEmrResourcePeriod"
      },
      "name": "MyOutputData1",
      "directoryPath": "s3://example-bucket/PigActivityOutput1",
      "dataFormat": {
        "ref": "MyOutputDataType1"
```

```
      },
      "type": "S3DataNode"
    },
    {
      "id": "MyInputDataType1",
      "name": "MyInputDataType1",
      "column": [
        "First STRING",
        "Second STRING",
        "Third STRING",
        "Fourth STRING",
        "Fifth STRING",
        "Sixth STRING",
        "Seventh STRING",
        "Eighth STRING",
        "Ninth STRING",
        "Tenth STRING"
      ],
      "inputRegEx": "^(\\S+) (\\S+) (\\S+) (\\S+) (\\S+) (\\S+) (\\S+) (\
\S+) (\\S+) (\\S+)",
      "type": "RegEx"
    },
    {
      "id": "MyEmrResource",
      "region": "us-east-1",
      "schedule": {
        "ref": "MyEmrResourcePeriod"
      },
      "keyPair": "example-keypair",
      "masterInstanceType": "m1.small",
      "enableDebugging": "true",
      "name": "MyEmrResource",
      "actionOnTaskFailure": "continue",
      "type": "EmrCluster"
    },
    {
      "id": "MyOutputDataType4",
      "name": "MyOutputDataType4",
      "column": "one STRING",
      "type": "CSV"
    },
    {
      "id": "MyOutputData4",
      "schedule": {
```

```
        "ref": "MyEmrResourcePeriod"
      },
      "directoryPath": "s3://example-bucket/PigActivityOutput3",
      "name": "MyOutputData4",
      "dataFormat": {
        "ref": "MyOutputDataType4"
      },
      "type": "S3DataNode"
    },
    {
      "id": "MyOutputDataType1",
      "name": "MyOutputDataType1",
      "column": [
        "First STRING",
        "Second STRING",
        "Third STRING",
        "Fourth STRING",
        "Fifth STRING",
        "Sixth STRING",
        "Seventh STRING",
        "Eighth STRING"
      ],
      "columnSeparator": "*",
      "type": "Custom"
    },
    {
      "id": "MyOutputData3",
      "username": "___",
      "schedule": {
        "ref": "MyEmrResourcePeriod"
      },
      "insertQuery": "insert into #{table} (one) values (?)",
      "name": "MyOutputData3",
      "*password": "___",
      "runsOn": {
        "ref": "MyEmrResource"
      },
      "connectionString": "jdbc:mysql://example-database-instance:3306/example-
database",
      "selectQuery": "select * from #{table}",
      "table": "example-table-name",
      "type": "MySqlDataNode"
    },
    {
```

```
      "id": "MyOutputDataType2",
      "name": "MyOutputDataType2",
      "column": [
        "Third STRING",
        "Fourth STRING",
        "Fifth STRING",
        "Sixth STRING",
        "Seventh STRING",
        "Eighth STRING"
      ],
      "type": "TSV"
    },
    {
      "id": "MyPigActivity2",
      "scheduleType": "CRON",
      "schedule": {
        "ref": "MyEmrResourcePeriod"
      },
      "input": {
        "ref": "MyOutputData1"
      },
      "pipelineLogUri": "s3://example-bucket/path",
      "name": "MyPigActivity2",
      "runsOn": {
        "ref": "MyEmrResource"
      },
      "dependsOn": {
        "ref": "MyPigActivity1"
      },
      "type": "PigActivity",
      "script": "B = LIMIT ${input1} 3; ${output1} = FOREACH B GENERATE Third, Fourth,
Fifth, Sixth, Seventh, Eighth;",
      "output": {
        "ref": "MyOutputData2"
      },
      "stage": "true"
    },
    {
      "id": "MyEmrResourcePeriod",
      "startDateTime": "2013-05-20T00:00:00",
      "name": "MyEmrResourcePeriod",
      "period": "1 day",
      "type": "Schedule",
      "endDateTime": "2013-05-21T00:00:00"
```

```
    },
    {
      "id": "MyPigActivity1",
      "scheduleType": "CRON",
      "schedule": {
        "ref": "MyEmrResourcePeriod"
      },
      "input": {
        "ref": "MyInputData1"
      },
      "pipelineLogUri": "s3://example-bucket/path",
      "scriptUri": "s3://example-bucket/script/pigTestScipt.q",
      "name": "MyPigActivity1",
      "runsOn": {
        "ref": "MyEmrResource"
      },
      "scriptVariable": [
        "column1=First",
        "column2=Second",
        "three=3"
      ],
      "type": "PigActivity",
      "output": {
        "ref": "MyOutputData1"
      },
      "stage": "true"
    }
  ]
}
```

pigTestScript.q 的内容如下所示。

```
B = LIMIT ${input1} $three; ${output1} = FOREACH B GENERATE $column1, $column2, Third,
 Fourth, Fifth, Sixth, Seventh, Eighth;
```

## 语法

| 对象调用字段 | 描述 | 槽类型 |
| --- | --- | --- |
| 计划 | 该对象在计划间隔的执行中调用。用户必须指定对另一个对象的计划引用，以便设置该对象的依赖项执行顺序。用户可在对象上明确设置 | 引用对象，例如，"schedule":{"ref":"mScheduleId"} |

| 对象调用字段 | 描述 | 槽类型 |
| --- | --- | --- |
| | 计划来满足此要求，例如，指定 "schedule": {"ref": "DefaultSchedule"}。在大多数情况下，最好将计划引用放在默认管道对象上，以便所有对象继承该计划。或者，如果管道有一个计划树 (计划位于主计划中)，用户可以创建具有计划引用的父对象。有关示例可选计划配置的更多信息，请参阅 https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html。 | |

| 所需的组 (下列选项之一是必需的) | 描述 | 槽类型 |
| --- | --- | --- |
| script | 要运行的 Pig 脚本。 | String |
| scriptUri | 要运行的 Pig 脚本的位置 (例如，s3://scriptLocation)。 | String |

| 所需的组 (下列选项之一是必需的) | 描述 | 槽类型 |
| --- | --- | --- |
| runsOn | 在其上运行此 PigActivity 的 EMR 集群。 | 引用对象，例如，"runsOn":{"ref":"myEmrClusterId"} |
| workerGroup | 工作线程组。这可用于路由任务。如果您提供 `runsOn` 值并且存在 `workerGroup` ，则将忽略 `workerGroup` | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| attemptStatus | 来自远程活动的最近报告的状态。 | String |
| attemptTimeout | 远程工作完成的超时时间。如果设置此字段，则可能会重试未在设定的开始时间内完成的远程活动。 | 周期 |
| dependsOn | 指定与另一个可运行对象的依赖关系。 | 引用对象，例如，"dependsOn":{"ref":"myActivityId"} |
| failureAndRerunMode | 描述依赖项失败或重新运行时的使用者节点行为。 | 枚举 |
| input | 输入数据源。 | 引用对象，例如，"input":{"ref":"myDataNodeId"} |
| lateAfterTimeout | 管道启动后经过的时间，在此时间内，对象必须完成。仅当计划类型未设置为 ondemand 时才会触发。 | 周期 |
| maxActiveInstances | 组件的并发活动实例的最大数量。重新运行不计入活动实例数中。 | 整数 |
| maximumRetries | 失败后的最大重试次数。 | 整数 |
| onFail | 当前对象失败时要运行的操作。 | 引用对象，例如，"onFail":{"ref":"myActionId"} |
| onLateAction | 在尚未计划对象或对象仍未完成的情况下将触发的操作。 | 引用对象，例如，"onLateAction":{"ref":"myActionId"} |
| onSuccess | 当前对象成功时要运行的操作。 | 引用对象，例如，"onSuccess":{"ref":"myActionId"} |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| output | 输出数据源。 | 引用对象，例如，"output":{"ref":"myDataNodeId"} |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |
| pipelineLogUri | 用于上传管道日志的 Amazon S3 URI (例如“s3://BucketName/Key/”)。 | String |
| postActivityTaskConfig | 要运行的活动后配置脚本。这由 Amazon S3 中 Shell 脚本的 URI 和参数列表组成。 | 引用对象，例如，"postActivityTaskConfig":{"ref":"myShellScriptConfigId"} |
| preActivityTaskConfig | 要运行的活动前配置脚本。这由 Amazon S3 中 Shell 脚本的 URI 和参数列表组成。 | 引用对象，例如，"preActivityTaskConfig":{"ref":"myShellScriptConfigId"} |
| precondition | (可选) 定义先决条件。在满足所有先决条件之前，数据节点不会标记为“READY”。 | 引用对象，例如，"precondition":{"ref":"myPreconditionId"} |
| reportProgressTimeout | 远程工作对 `reportProgress` 的连续调用的超时时间。如果设置此字段，则未报告指定时段的进度的远程活动可能会被视为停滞且已重试。 | 周期 |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| resizeClusterBeforeRunning | 在执行此活动前，重新调整集群的大小，以适应指定为输入或输出的 DynamoDB 数据节点。<br><br>**Note**<br>如果您的活动使用 DynamoDBDataNode 作为输入或输出数据节点，而且您将 resizeClusterBeforeRunning 设置为 TRUE，AWS Data Pipeline 将开始使用 m3.xlarge 实例类型。这将使用 m3.xlarge 覆盖您的实例类型，从而可能会增加您的月度成本。 | 布尔值 |
| resizeClusterMaxInstances | 调整大小算法可以请求的最大实例数的限制。 | 整数 |
| retryDelay | 两次重试之间的超时时间。 | 周期 |
| scheduleType | 计划类型允许您指定应在间隔的结尾还是开头计划您管道定义中的对象。时间序列风格计划表示在每次间隔的结尾计划实例，而 Cron 风格计划表示应在每次间隔的开头计划实例。按需计划让您可以在每次激活时运行一次管道。这意味着，您不需要克隆或重新创建管道以再次运行它。如果您使用按需计划，则必须在默认对象中指定它，并且必须是在管道中为对象指定的唯一 scheduleType。要使用按需管道，您只需为每个后续运行调用 ActivatePipeline 操作。值包括：cron、ondemand 和 timeseries。 | 枚举 |
| scriptVariable | 要传递到 Pig 脚本的参数。您可以将 scriptVariable 与 script 或 scriptUri 一起使用。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| stage | 确定是否启用了暂存并且您的 Pig 脚本有权访问暂存数据表，例如 ${INPUT1} 和 ${OUTPUT1}。 | 布尔值 |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @activeInstances | 当前计划的有效实例对象的列表。 | 引用对象，例如，"activeInstances":{"ref":"myRunnableObjectId"} |
| @actualEndTime | 该对象的执行完成时间。 | DateTime |
| @actualStartTime | 该对象的执行开始时间。 | DateTime |
| cancellationReason | 该对象被取消时显示的 cancellationReason。 | String |
| @cascadeFailedOn | 对象在其上失败的依赖项链的描述。 | 引用对象，例如，"cascadeFailedOn":{"ref":"myRunnableObjectId"} |
| emrStepLog | 仅在尝试 EMR 活动时可用的 Amazon EMR 步骤日志。 | String |
| errorId | 该对象失败时显示的 errorId。 | String |
| errorMessage | 该对象失败时显示的 errorMessage。 | String |
| errorStackTrace | 该对象失败时显示的错误堆栈跟踪。 | String |
| @finishedTime | 该对象完成其执行的时间。 | DateTime |
| hadoopJobLog | 在尝试基于 EMR 的活动时可用的 Hadoop 任务日志。 | String |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @healthStatus | 对象的运行状况，反映进入终止状态的上个对象实例成功还是失败。 | String |
| @healthStatusFromInstanceId | 进入终止状态的上个实例对象的 ID。 | String |
| @healthStatusUpdatedTime | 上次更新运行状况的时间。 | DateTime |
| hostname | 已执行任务尝试的客户端的主机名。 | String |
| @lastDeactivatedTime | 上次停用该对象的时间。 | DateTime |
| @latestCompletedRunTime | 已完成执行的最新运行的时间。 | DateTime |
| @latestRunTime | 已计划执行的最新运行的时间。 | DateTime |
| @nextRunTime | 计划下次运行的时间。 | DateTime |
| reportProgressTime | 远程活动报告进度的最近时间。 | DateTime |
| @scheduledEndTime | 对象的计划结束时间。 | DateTime |
| @scheduledStartTime | 对象的计划开始时间。 | DateTime |
| @status | 该对象的状态。 | String |
| @version | 用于创建对象的管道版本。 | String |
| @waitingOn | 该对象在其上处于等待状态的依赖项列表的描述。 | 引用对象，例如，"waitingOn":{"ref":"myRunnableObjectId"} |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息。 | String |
| @pipelineId | 该对象所属的管道的 ID。 | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象。 | String |

### 另请参阅

- ShellCommandActivity
- EmrActivity

## RedshiftCopyActivity

将数据从 DynamoDB 或 Amazon S3 复制到 Amazon Redshift。您可以将数据加载到新表中，或轻松地将数据并入现有表中。

下面概括了使用 `RedshiftCopyActivity` 的使用案例：

1. 开始使用 AWS Data Pipeline 在 Amazon S3 中存放您的数据。
2. 使用 `RedshiftCopyActivity` 将数据从 Amazon RDS 和 Amazon EMR 移动到 Amazon Redshift。

   这可让您将数据加载到 Amazon Redshift 中，您可以在此处对数据进行分析。
3. 使用 SqlActivity 可以对已加载到 Amazon Redshift 中的数据执行 SQL 查询。

此外，借助 `RedshiftCopyActivity`，您可以使用 `S3DataNode`，因为它支持清单文件。有关更多信息，请参阅S3DataNode。

### 示例

以下是该对象类型的示例。

为了确保格式转换，本示例在 `commandOptions` 中使用 EMPTYASNULL 和 IGNOREBLANKLINES 特殊转换参数。有关信息，请参阅 Amazon Redshift 数据库开发人员指南中的数据转换参数。

```
{
  "id" : "S3ToRedshiftCopyActivity",
  "type" : "RedshiftCopyActivity",
  "input" : { "ref": "MyS3DataNode" },
  "output" : { "ref": "MyRedshiftDataNode" },
  "insertMode" : "KEEP_EXISTING",
  "schedule" : { "ref": "Hour" },
  "runsOn" : { "ref": "MyEc2Resource" },
  "commandOptions": ["EMPTYASNULL", "IGNOREBLANKLINES"]
}
```

以下示例管道定义说明了一个使用 APPEND 插入模式的活动：

```
{
  "objects": [
    {
      "id": "CSVId1",
      "name": "DefaultCSV1",
      "type": "CSV"
    },
    {
      "id": "RedshiftDatabaseId1",
      "databaseName": "dbname",
      "username": "user",
      "name": "DefaultRedshiftDatabase1",
      "*password": "password",
      "type": "RedshiftDatabase",
      "clusterId": "redshiftclusterId"
    },
    {
      "id": "Default",
      "scheduleType": "timeseries",
      "failureAndRerunMode": "CASCADE",
      "name": "Default",
      "role": "DataPipelineDefaultRole",
      "resourceRole": "DataPipelineDefaultResourceRole"
    },
    {
      "id": "RedshiftDataNodeId1",
      "schedule": {
        "ref": "ScheduleId1"
      },
      "tableName": "orders",
```

```
      "name": "DefaultRedshiftDataNode1",
      "createTableSql": "create table StructuredLogs (requestBeginTime CHAR(30)
PRIMARY KEY DISTKEY SORTKEY, requestEndTime CHAR(30), hostname CHAR(100), requestDate
varchar(20));",
      "type": "RedshiftDataNode",
      "database": {
        "ref": "RedshiftDatabaseId1"
      }
    },
    {
      "id": "Ec2ResourceId1",
      "schedule": {
        "ref": "ScheduleId1"
      },
      "securityGroups": "MySecurityGroup",
      "name": "DefaultEc2Resource1",
      "role": "DataPipelineDefaultRole",
      "logUri": "s3://myLogs",
      "resourceRole": "DataPipelineDefaultResourceRole",
      "type": "Ec2Resource"
    },
    {
      "id": "ScheduleId1",
      "startDateTime": "yyyy-mm-ddT00:00:00",
      "name": "DefaultSchedule1",
      "type": "Schedule",
      "period": "period",
      "endDateTime": "yyyy-mm-ddT00:00:00"
    },
    {
      "id": "S3DataNodeId1",
      "schedule": {
        "ref": "ScheduleId1"
      },
      "filePath": "s3://datapipeline-us-east-1/samples/hive-ads-samples.csv",
      "name": "DefaultS3DataNode1",
      "dataFormat": {
        "ref": "CSVId1"
      },
      "type": "S3DataNode"
    },
    {
      "id": "RedshiftCopyActivityId1",
      "input": {
```

```
        "ref": "S3DataNodeId1"
      },
      "schedule": {
        "ref": "ScheduleId1"
      },
      "insertMode": "APPEND",
      "name": "DefaultRedshiftCopyActivity1",
      "runsOn": {
        "ref": "Ec2ResourceId1"
      },
      "type": "RedshiftCopyActivity",
      "output": {
        "ref": "RedshiftDataNodeId1"
      }
    }
  ]
}
```

APPEND 操作向表中添加项，无论主键或排序键如何。例如，如果您有以下表，则可追加具有相同的 ID 和用户值的记录。

```
ID(PK)     USER
1          aaa
2          bbb
```

您可以追加具有相同的 ID 和用户值的记录：

```
ID(PK)     USER
1          aaa
2          bbb
1          aaa
```

> **Note**
>
> 如果 APPEND 操作中断并重试，生成的重新运行管道可能会从开始位置追加。这可能会导致进一步复制，因此，您应了解此行为，尤其是当您有任何计算行数的逻辑时。

有关教程，请参阅 使用 AWS Data Pipeline 复制数据到 Amazon Redshift。

## 语法

| 必填字段 | 描述 | 槽类型 |
| --- | --- | --- |
| insertMode | 确定在目标表中预先存在的数据与要加载数据中的行重叠时，AWS Data Pipeline 执行什么操作。<br><br>有效值包括：KEEP_EXISTING 、OVERWRITE_EXISTING 、TRUNCATE 和 APPEND。<br><br>KEEP_EXISTING 添加新行到表中，同时保留任何现有的行不变。<br><br>KEEP_EXISTING 和 OVERWRITE_EXISTING 使用主键、排序键和分配键来识别哪些传入行与现有行匹配。请参阅 Amazon Redshift 数据库开发人员指南中的更新和插入新数据。<br><br>TRUNCATE 先删除目标表中的所有数据，然后写入新数据。<br><br>APPEND 会将所有记录添加到 Redshift 表的结尾。APPEND 不需要主键、分配键或排序键，因此会附加可能存在重复的项。 | 枚举 |

| 对象调用字段 | 描述 | 槽类型 |
| --- | --- | --- |
| 计划 | 该对象在计划间隔的执行中调用。<br><br>指定对另一个对象的计划引用，以便设置该对象的依赖项执行顺序。<br><br>在大多数情况下，我们建议将计划引用放在默认管道对象上，以便所有对象继承该计划。例如，您可以通过指定 "schedule": {"ref": | 引用对象，例如： "schedule":{"ref": "myScheduleId"} |

| 对象调用字段 | 描述 | 槽类型 |
| --- | --- | --- |
| | "DefaultSchedule"} ，明确地针对该对象设置计划。<br><br>如果您的管道中的主计划包含嵌套计划，则可以创建具有计划引用的父对象。<br><br>有关示例可选计划配置的更多信息，请参阅计划。 | |

| 所需的组 (下列选项之一是必需的) | 描述 | 槽类型 |
| --- | --- | --- |
| runsOn | 运行活动或命令的计算资源。例如，Amazon EC2 实例或 Amazon EMR 集群。 | 引用对象，例如，"runsOn":{"ref":"myResourceId"} |
| workerGroup | 工作线程组。这可用于路由任务。如果您提供 runsOn 值并且存在 workerGroup ，则将忽略 workerGroup。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| attemptStatus | 来自远程活动的最近报告的状态。 | String |
| attemptTimeout | 远程工作完成的超时时间。如果设置此字段，则可能会重试未在设定的开始时间内完成的远程活动。 | 周期 |
| commandOptions | 获取在 COPY 操作期间传递到 Amazon Redshift 数据节点的参数。有关更多信息，请参阅 Amazon Redshift 数据库开发人员指南中的 COPY。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| | 在加载表时，COPY 会尝试将字符串隐式转换为目标列的数据类型。如果您收到错误或有其他转换需求，则除了自动发生的默认数据转换之外，您还可以指定其他转换参数。有关信息，请参阅 Amazon Redshift 数据库开发人员指南中的数据转换参数。<br><br>如果数据格式与输入或输出数据节点关联，则忽略提供的参数。<br><br>由于复制操作首先使用 COPY 将数据插入暂存表，然后使用 INSERT 命令将数据从暂存表复制到目标表中，一些 COPY 参数不适用，例如 COPY 命令启用表上自动压缩的功能。如果需要压缩，请向 CREATE TABLE 语句添加列编码详细信息。<br><br>此外，在某些需要从 Amazon Redshift 集群卸载数据和在 Amazon S3 中创建文件的情况下，RedshiftCopyActivity 依赖 Amazon Redshift 的 UNLOAD 操作。<br><br>为提高复制和卸载过程中的性能，请从 UNLOAD 命令指定 PARALLEL OFF 参数。有关更多信息，请参阅 Amazon Redshift 数据库开发人员指南中的 UNLOAD。 | |
| dependsOn | 指定与另一个可运行对象的依赖关系。 | 引用对象：`"dependsOn":{"ref":"myActivityId"}` |
| failureAndRerunMode | 描述依赖项失败或重新运行时的使用者节点行为。 | 枚举 |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| input | 输入数据节点。数据源可以是 Amazon S3、DynamoDB 或 Amazon Redshift。 | 引用对象：`"input":{"ref":"myDataNodeId"}` |
| lateAfterTimeout | 管道启动后经过的时间，在此时间内，对象必须完成。仅当计划类型未设置为 ondemand 时才会触发。 | 周期 |
| maxActiveInstances | 组件的并发活动实例的最大数量。重新运行不计入活动实例数中。 | 整数 |
| maximumRetries | 失败后的最大重试次数 | 整数 |
| onFail | 当前对象失败时要运行的操作。 | 引用对象：`"onFail":{"ref":"myActionId"}` |
| onLateAction | 在尚未计划对象或对象仍未完成的情况下将触发的操作。 | 引用对象：`"onLateAction":{"ref":"myActionId"}` |
| onSuccess | 当前对象成功时要运行的操作。 | 引用对象：`"onSuccess":{"ref":"myActionId"}` |
| output | 输出数据节点。输出位置可以是 Amazon S3 或 Amazon Redshift。 | 引用对象：`"output":{"ref":"myDataNodeId"}` |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| parent | 槽将继承自的当前对象的父级。 | 引用对象：`"parent":{"ref":"myBaseObjectId"}` |
| pipelineLogUri | 用于上传管道日志的 S3 URI (例如“s3://BucketName/Key/”)。 | String |
| precondition | (可选) 定义先决条件。在满足所有先决条件之前，数据节点不会标记为“READY”。 | 引用对象：`"precondition":{"ref":"myPreconditionId"}` |
| 队列 | 对应于 Amazon Redshift 中的 `query_group` 设置，这允许您根据放置在队列中的位置分配并优先处理并发活动。<br><br>Amazon Redshift 将同时连接的数量限制为 15。有关更多信息，请参阅 Amazon RDS 数据库开发人员指南中的向队列分配查询。 | String |
| reportProgressTimeout | 远程工作对 `reportProgress` 的连续调用的超时时间。<br><br>如果设置此字段，则未报告指定时段的进度的远程活动可能会被视为停滞且已重试。 | 周期 |
| retryDelay | 两次重试之间的超时时间。 | 周期 |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| scheduleType | 允许您指定是否计划管道中的对象。值包括：cron、ondemand 和 timeseries 。<br><br>timeseries 计划表示在每个间隔结束时计划实例。<br><br>Cron 计划表示在每个间隔开始时计划实例。<br><br>ondemand 计划让您可以在每次激活时运行一次管道。这意味着，您不需要克隆或重新创建管道以再次运行它。<br><br>要使用 ondemand 管道，请为每个后续运行调用 ActivatePipeline 操作。<br><br>如果您使用 ondemand 计划，您必须在默认对象中指定它，并且该计划必须是在管道中为对象指定的唯一 scheduleType 。 | 枚举 |
| transformSql | 用于转换输入数据的 SQL SELECT 表达式。<br><br>在名为 staging 的表上运行 transformSql 表达式。<br><br>当您从 DynamoDB 或 Amazon S3 复制数据时，AWS Data Pipeline 会创建一个名为“staging”的表，并且最初在该表中加载数据。此表中的数据用于更新目标表。<br><br>transformSql 的输出架构必须与最终目标表的架构匹配。<br><br>如果您指定了 transformSql 选项，则会从指定的 SQL 语句创建第二个暂存表。然后，来自这第二个暂存表的数据更新到最终的目标表中。 | String |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @activeInstances | 当前计划的有效实例对象的列表。 | 引用对象："activeInstances":{"ref":"myRunnableObjectId"} |
| @actualEndTime | 该对象的执行完成时间。 | DateTime |
| @actualStartTime | 该对象的执行开始时间。 | DateTime |
| cancellationReason | 该对象被取消时显示的 cancellationReason。 | String |
| @cascadeFailedOn | 对象在其上失败的依赖项链的描述。 | 引用对象："cascadeFailedOn":{"ref":"myRunnableObjectId"} |
| emrStepLog | 仅在尝试 EMR 活动时可用的 EMR 步骤日志 | String |
| errorId | 该对象失败时显示的 errorId。 | String |
| errorMessage | 该对象失败时显示的 errorMessage。 | String |
| errorStackTrace | 该对象失败时显示的错误堆栈跟踪。 | String |
| @finishedTime | 该对象完成其执行的时间。 | DateTime |
| hadoopJobLog | 在尝试基于 EMR 的活动时可用的 Hadoop 任务日志。 | String |
| @healthStatus | 对象的运行状况，反映进入终止状态的上个对象实例成功还是失败。 | String |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @healthStatusFromInstanceId | 进入终止状态的上个实例对象的 ID。 | String |
| @healthStatusUpdatedTime | 上次更新运行状况的时间。 | DateTime |
| hostname | 已执行任务尝试的客户端的主机名。 | String |
| @lastDeactivatedTime | 上次停用该对象的时间。 | DateTime |
| @latestCompletedRunTime | 已完成执行的最新运行的时间。 | DateTime |
| @latestRunTime | 已计划执行的最新运行的时间。 | DateTime |
| @nextRunTime | 计划下次运行的时间。 | DateTime |
| reportProgressTime | 远程活动报告进度的最近时间。 | DateTime |
| @scheduledEndTime | 对象的计划结束时间。 | DateTime |
| @scheduledStartTime | 对象的计划开始时间。 | DateTime |
| @status | 该对象的状态。 | String |
| @version | 用来创建对象的管道版本。 | String |
| @waitingOn | 该对象在其上处于等待状态的依赖项列表的描述。 | 引用对象：`"waitingOn": {"ref":"myRunnableObjectId"}` |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息。 | String |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @pipelineId | 该对象所属的管道的 ID。 | String |
| @sphere | 对象的球体。表示对象在生命周期中的位置。例如，组件对象产生实例对象，后者执行尝试对象。 | String |

## ShellCommandActivity

运行命令或脚本。您可以使用 ShellCommandActivity 运行时间序列或类似 Cron 的计划任务。

当 stage 字段设置为 true 并与 S3DataNode 结合使用时，ShellCommandActivity 支持暂存数据概念，这意味着，您可以将数据从 Amazon S3 移至暂存位置（如 Amazon EC2）或您的本地环境，使用脚本和 ShellCommandActivity 对数据执行工作，并将数据移回 Amazon S3。

在这种情况下，当 Shell 命令连接到输入 S3DataNode 时，Shell 脚本使用 ${INPUT1_STAGING_DIR}、${INPUT2_STAGING_DIR} 及其他字段 (请参阅 ShellCommandActivity 输入字段) 直接操作数据。

同样，shell 命令输出可暂存到一个输出目录中，以便通过 ${OUTPUT1_STAGING_DIR}、${OUTPUT2_STAGING_DIR} 等引用的方式自动推送到 Amazon S3。

这些表达式可作为命令行参数传递到 shell 命令，以供您在数据转换逻辑中使用。

ShellCommandActivity 返回 Linux 样式的错误代码和字符串。如果 ShellCommandActivity 生成错误，则返回的 error 为非零值。

### 示例

以下是该对象类型的示例。

```
{
  "id" : "CreateDirectory",
  "type" : "ShellCommandActivity",
  "command" : "mkdir new-directory"
}
```

## 语法

| 对象调用字段 | 描述 | 槽类型 |
| --- | --- | --- |
| 计划 | 该对象在 schedule 间隔的执行中调用。<br><br>要设置该对象的依赖项执行顺序，请指定对另一个对象的 schedule 引用。<br><br>为满足此要求，可在对象上明确设置 schedule，例如，指定 "schedule": {"ref": "DefaultSchedule"} 。<br><br>在大多数情况下，最好将 schedule 引用放在默认管道对象上，以便所有对象继承该计划。如果管道包含计划树 (计划位于主计划中)，可以创建具有计划引用的父对象。<br><br>为了分配负载，AWS Data Pipeline 稍微提前于计划创建物理对象，但在计划期间运行它们。<br><br>有关示例可选计划配置的更多信息，请参阅 https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html。 | 引用对象，例如，"schedule":{"ref":"myScheduleId"} |

| 所需的组 (下列选项之一是必需的) | 描述 | 槽类型 |
| --- | --- | --- |
| 命令 | 要运行的命令。使用 $ 引用位置参数，使用 scriptArgument 指定命令的参数。此值与任何关联参数必须在从中运行任务运行程序的环境中起作用。 | String |
| scriptUri | 要下载并作为 shell 命令运行的文件的 Amazon S3 URI 路径。仅指定一个 scriptUri 或 | String |

| 所需的组 (下列选项之一是必需的) | 描述 | 槽类型 |
| --- | --- | --- |
| | command 字段。scriptUri 不能使用参数，请使用 command。 | |

| 所需的组 (下列选项之一是必需的) | 描述 | 槽类型 |
| --- | --- | --- |
| runsOn | 运行活动或命令的计算资源，例如 Amazon EC2 实例或 Amazon EMR 集群。 | 引用对象，例如，"runsOn":{"ref":"myResourceId"} |
| workerGroup | 用于路由任务。如果您提供 runs0n 值并且存在 workerGroup ，则将忽略 workerGroup | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| attemptStatus | 来自远程活动的最近报告的状态。 | String |
| attemptTimeout | 远程工作完成的超时时间。如果设置此字段，则可能会重试未在指定开始时间内完成的远程活动。 | 周期 |
| dependsOn | 指定与另一个可运行对象的依赖关系。 | 引用对象，例如，"dependsOn":{"ref":"myActivityId"} |
| failureAndRerunMode | 描述依赖项失败或重新运行时的使用者节点行为。 | 枚举 |
| input | 输入数据的位置。 | 引用对象，例如，"input":{"ref":"myDataNodeId"} |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| lateAfterTimeout | 管道启动后经过的时间，在此时间内，对象必须完成。仅当计划类型未设置为 ondemand 时才会触发。 | 周期 |
| maxActiveInstances | 组件的并发活动实例的最大数量。重新运行不计入活动实例数中。 | 整数 |
| maximumRetries | 失败后的最大重试次数。 | 整数 |
| onFail | 当前对象失败时要运行的操作。 | 引用对象，例如，"onFail":{"ref":"myActionId"} |
| onLateAction | 在尚未计划对象或对象未完成的情况下将触发的操作。 | 引用对象，例如，"onLateAction":{"ref":"myActionId"} |
| onSuccess | 当前对象成功时要运行的操作。 | 引用对象，例如，"onSuccess":{"ref":"myActionId"} |
| output | 输出数据的位置。 | 引用对象，例如，"output":{"ref":"myDataNodeId"} |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |
| pipelineLogUri | 用于上传管道日志的 Amazon S3 URI，例如 's3://BucketName/Key/' 。 | String |
| precondition | (可选) 定义先决条件。在满足所有先决条件之前，数据节点不会标记为“READY”。 | 引用对象，例如，"precondition":{"ref":"myPreconditionId"} |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| reportProgressTimeout | 远程活动对 reportProgress 的连续调用的超时时间。如果设置此字段，则未报告指定时段的进度的远程活动可能会被视为停滞且已重试。 | 周期 |
| retryDelay | 两次重试之间的超时时间。 | 周期 |
| scheduleType | 允许您指定应在间隔的开头还是结尾计划您管道定义中的对象。<br><br>值为 cron、ondemand 和 timeseries 。<br><br>如果设置为 timeseries ，则在每个间隔结束时计划实例。<br><br>如果设置为 Cron，则在每个间隔开始时计划实例。<br><br>如果设置为 ondemand，您可以在每次激活时运行一次管道。这意味着，您不需要克隆或重新创建管道以再次运行它。如果使用 ondemand 计划，则在默认对象中指定它，并且该计划必须是在管道中为对象指定的唯一 scheduleType 。要使用 ondemand 管道，请为每个后续运行调用 ActivatePipeline 操作。 | 枚举 |
| scriptArgument | 传递到命令所指定命令的 JSON 格式的字符串数组。例如，如果命令是 echo $1 $2，请将 scriptArgument 指定为 "param1", "param2"。对于多个自变量和参数，按如下方式传递 scriptArgument ： "scriptArgument":"arg1","scriptArgument":"param1","scriptArgument":"arg2","scriptArgument":"param2" 。scriptArgument 只能与 command 一起使用；将其与 scriptUri 一起使用会导致错误。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| stage | 确定是否启用了暂存并且 Shell 命令有权访问暂存数据变量，例如 `${INPUT1_STAGING_DIR}` 和 `${OUTPUT1_STAGING_DIR}` 。 | 布尔值 |
| stderr | 接收来自命令的重定向系统错误消息的 路径。如果您使用 `runsOn` 字段，则由于运行活动的资源的短期性质，该字段必须为 Amazon S3 路径。不过，如果指定 `workerGroup` 字段，则允许使用本地文件路径。 | String |
| stdout | 接收来自命令的重定向输出的 Amazon S3 路径。如果您使用 `runsOn` 字段，则由于运行活动的资源的短期性质，该字段必须为 Amazon S3 路径。不过，如果指定 `workerGroup` 字段，则允许使用本地文件路径。 | String |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @activeInstances | 当前计划的有效实例对象的列表。 | 引用对象，例如，"activeInstances":{"ref":"myRunnableObjectId"} |
| @actualEndTime | 该对象的执行完成时间。 | DateTime |
| @actualStartTime | 该对象的执行开始时间。 | DateTime |
| cancellationReason | 该对象被取消时显示的 `cancellationReason` 。 | String |
| @cascadeFailedOn | 导致对象失败的依赖项链的描述。 | 引用对象，例如，"cascadeFailedOn":{"ref":"myRunnableObjectId"} |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| emrStepLog | 仅在尝试 Amazon EMR 活动时可用的 Amazon EMR 步骤日志 | String |
| errorId | 该对象失败时显示的 `errorId`。 | String |
| errorMessage | 该对象失败时显示的 `errorMessage` 。 | String |
| errorStackTrace | 该对象失败时显示的错误堆栈跟踪。 | String |
| @finishedTime | 对象完成其执行的时间。 | DateTime |
| hadoopJobLog | 在尝试基于 Amazon EMR 的活动时可用的 Hadoop 任务日志。 | String |
| @healthStatus | 对象的运行状况，反映进入终止状态的上个对象实例成功还是失败。 | String |
| @healthStatusFromInstanceId | 进入终止状态的上个实例对象的 ID。 | String |
| @healthStatusUpdatedTime | 上次更新运行状况的时间。 | DateTime |
| hostname | 已执行任务尝试的客户端的主机名。 | String |
| @lastDeactivatedTime | 上次停用该对象的时间。 | DateTime |
| @latestCompletedRunTime | 已完成执行的最新运行的时间。 | DateTime |
| @latestRunTime | 已计划执行的最新运行的时间。 | DateTime |
| @nextRunTime | 计划下次运行的时间。 | DateTime |
| reportProgressTime | 远程活动报告进度的最近时间。 | DateTime |
| @scheduledEndTime | 对象的计划结束时间。 | DateTime |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @scheduledStartTime | 对象的计划开始时间。 | DateTime |
| @status | 对象的状态。 | String |
| @version | 用于创建对象的 AWS Data Pipeline 版本。 | String |
| @waitingOn | 该对象等待的依赖项列表的描述。 | 引用对象，例如，"waitingOn":{"ref": "myRunnableObjectId"} |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息。 | String |
| @pipelineId | 该对象所属的管道的 ID。 | String |
| @sphere | 对象在生命周期中的位置。组件对象产生实例对象，后者执行尝试对象。 | String |

### 另请参阅

- CopyActivity
- EmrActivity

## SqlActivity

在数据库上运行 SQL 查询 (脚本)。

### 示例

以下是该对象类型的示例。

```
{
```

```
  "id" : "MySqlActivity",
  "type" : "SqlActivity",
  "database" : { "ref": "MyDatabaseID" },
  "script" : "SQLQuery" | "scriptUri" : s3://scriptBucket/query.sql,
  "schedule" : { "ref": "MyScheduleID" },
}
```

## 语法

| 必填字段 | 描述 | 槽类型 |
| --- | --- | --- |
| 数据库 | 要在其上运行提供的 SQL 脚本的数据库。 | 引用对象，例如，"database":{"ref":"myDatabaseId"} |

| 对象调用字段 | 描述 | 槽类型 |
| --- | --- | --- |
| 计划 | 该对象在计划间隔的执行中调用。您必须指定对另一个对象的计划引用，以便设置该对象的依赖项执行顺序。您可以明确地针对该对象设置计划，例如通过指定 `"schedule": {"ref": "DefaultSchedule"}` 。<br><br>在大多数情况下，最好将计划引用放在默认管道对象上，以便所有对象继承该计划。<br><br>如果管道有一个嵌套在主计划中的计划树，则可以创建具有计划引用的父对象。有关示例可选计划配置的更多信息，请参阅 https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html。 | 引用对象，例如，"schedule":{"ref":"myScheduleId"} |

| 所需的组 (下列选项之一是必需的) | 描述 | 槽类型 |
| --- | --- | --- |
| script | 要运行的 SQL 脚本。您必须指定 script 或 scriptUri。在将脚本存储在 Amazon S3 中时，脚本不会计算为表达式。在将脚本存储在 Amazon S3 中时，为 scriptArgument 指定多个值会很有用。 | String |
| scriptUri | 一个 URI，指定要在此活动中执行的 SQL 脚本的位置。 | String |

| 所需的组 (下列选项之一是必需的) | 描述 | 槽类型 |
| --- | --- | --- |
| runsOn | 运行活动或命令的计算资源。例如，Amazon EC2 实例或 Amazon EMR 集群。 | 引用对象，例如，"runsOn":{"ref":"myResourceId"} |
| workerGroup | 工作线程组。这可用于路由任务。如果您提供 `runsOn` 值并且存在 `workerGroup` ，则将忽略 `workerGroup` | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| attemptStatus | 来自远程活动的最近报告的状态。 | String |
| attemptTimeout | 远程工作完成的超时时间。如果设置此字段，则可能会重试未在设定的开始时间内完成的远程活动。 | 周期 |
| dependsOn | 指定与另一个可运行对象的依赖关系。 | 引用对象，例如，"dependsOn":{"ref":"myActivityId"} |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| failureAndRerunMode | 描述依赖项失败或重新运行时的使用者节点行为。 | 枚举 |
| input | 输入数据的位置。 | 引用对象，例如，"input":{"ref":"myDataNodeId"} |
| lateAfterTimeout | 自管道的计划开始时间后的时段，对象运行必须在该时段内开始。 | 周期 |
| maxActiveInstances | 组件的并发活动实例的最大数量。重新运行不计入活动实例数中。 | 整数 |
| maximumRetries | 失败后的最大重试次数 | 整数 |
| onFail | 当前对象失败时要运行的操作。 | 引用对象，例如，"onFail":{"ref":"myActionId"} |
| onLateAction | 在自管道的计划开始时间后的时段 (由“lateAfterTimeout”指定) 内未计划对象或对象仍未完成时应触发的操作。 | 引用对象，例如，"onLateAction":{"ref":"myActionId"} |
| onSuccess | 当前对象成功时要运行的操作。 | 引用对象，例如，"onSuccess":{"ref":"myActionId"} |
| output | 输出数据的位置。这仅在从脚本 (例如，`#{output.tablename}` ) 中引用以及通过在输出数据节点中设置“createTableSql”来创建输出表时会很有用。SQL 查询的输出不会写入输出数据节点。 | 引用对象，例如，"output":{"ref":"myDataNodeId"} |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| pipelineLogUri | 用于上传管道日志的 S3 URI (例如“s3://BucketName/Key/”)。 | String |
| precondition | (可选) 定义先决条件。在满足所有先决条件之前，数据节点不会标记为“READY”。 | 引用对象，例如，"precondition":{"ref":"myPreconditionId"} |
| 队列 | [仅 Amazon Redshift] 对应于 Amazon Redshift 中的 query_group 设置，允许您根据在队列中的放置位置来分配并发活动以及确定它们的优先级。Amazon Redshift 将同时连接的数量限制为 15。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的向队列分配查询。 | String |
| reportProgressTimeout | 远程工作对 reportProgress 的连续调用的超时时间。如果设置此字段，则未报告指定时段的进度的远程活动可能会被视为停滞且已重试。 | 周期 |
| retryDelay | 两次重试之间的超时时间。 | 周期 |
| scheduleType | 计划类型允许您指定应在间隔的结尾还是开头计划您管道定义中的对象。值包括：cron、ondemand 和 timeseries 。<br><br>timeseries 计划表示在每个间隔结束时计划实例。<br><br>cron 计划表示在每个间隔开始时计划实例。<br><br>ondemand 计划让您可以在每次激活时运行一次管道。这意味着，您不需要克隆或重新创建管道以再次运行它。如果您使用 ondemand 计划，则必须在默认对象中指定它，并且该计划必须是在管道中为对象指定的唯一 scheduleType 。要使用 ondemand 管道，请为每个后续运行调用 ActivatePipeline 操作。 | 枚举 |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| scriptArgument | 脚本的变量列表。或者，您可以直接在脚本字段中放置表达式。在将脚本存储在 Amazon S3 中时，scriptArgument 的多个值会很有用。示例：#{format(@scheduledStartTime, "YY-MM-DD HH:MM:SS"}\n#{format(plusPeriod(@scheduledStartTime, "1 day"), "YY-MM-DD HH:MM:SS"} | String |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @activeInstances | 当前计划的有效实例对象的列表。 | 引用对象，例如，"activeInstances":{"ref":"myRunnableObjectId"} |
| @actualEndTime | 该对象的执行完成时间。 | DateTime |
| @actualStartTime | 该对象的执行开始时间。 | DateTime |
| cancellationReason | 该对象被取消时显示的 cancellationReason。 | String |
| @cascadeFailedOn | 对象在其上失败的依赖项链的描述。 | 引用对象，例如，"cascadeFailedOn":{"ref":"myRunnableObjectId"} |
| emrStepLog | 仅在尝试 EMR 活动时可用的 EMR 步骤日志 | String |
| errorId | 该对象失败时显示的 errorId。 | String |
| errorMessage | 该对象失败时显示的 errorMessage。 | String |
| errorStackTrace | 该对象失败时显示的错误堆栈跟踪。 | String |
| @finishedTime | 该对象完成其执行的时间。 | DateTime |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| hadoopJobLog | 在尝试基于 EMR 的活动时可用的 Hadoop 任务日志。 | String |
| @healthStatus | 对象的运行状况，反映进入终止状态的上个对象实例成功还是失败。 | String |
| @healthStatusFromInstanceId | 进入终止状态的上个实例对象的 ID。 | String |
| @healthStatusUpdatedTime | 上次更新运行状况的时间。 | DateTime |
| hostname | 已执行任务尝试的客户端的主机名。 | String |
| @lastDeactivatedTime | 上次停用该对象的时间。 | DateTime |
| @latestCompletedRunTime | 已完成执行的最新运行的时间。 | DateTime |
| @latestRunTime | 已计划执行的最新运行的时间。 | DateTime |
| @nextRunTime | 计划下次运行的时间。 | DateTime |
| reportProgressTime | 远程活动报告进度的最近时间。 | DateTime |
| @scheduledEndTime | 对象的计划结束时间。 | DateTime |
| @scheduledStartTime | 对象的计划开始时间。 | DateTime |
| @status | 该对象的状态。 | String |
| @version | 用来创建对象的管道版本。 | String |
| @waitingOn | 该对象在其上处于等待状态的依赖项列表的描述。 | 引用对象，例如，"waitingOn":{"ref": "myRunnableObjectId"} |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息。 | String |
| @pipelineId | 该对象所属的管道的 ID。 | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象。 | String |

# 资源

以下是 AWS Data Pipeline 资源对象：

Objects

- Ec2Resource
- EmrCluster
- HttpProxy

## Ec2Resource

执行管道活动定义的工作的 Amazon EC2 实例。

AWS Data Pipeline现在支持 Amazon EC2 实例的 imdsv2，当从实例检索元数据信息时，它使用面向会话的方法来更好地处理身份验证。会话会开始和结束一系列请求，Amazon EC2 实例上运行的软件使用这些请求访问本地存储的 Amazon EC2 实例元数据和凭证。该软件通过向 imdsv2 发出简单的 HTTP PUT 请求来开始会话。IMDSv2 向在 Amazon EC2 实例上运行的软件返回一个秘密令牌，该软件将使用该令牌作为密码向 IMDSv2 请求元数据和凭证。

> **Note**
>
> 要将 IMDSv2 用于您的 Amazon EC2 实例，您需要修改设置，因为默认 AMI 与 IMDSv2 不兼容。您可以指定一个新的 AMI 版本，您可以通过以下 SSM 参数检索该版本：`/aws/service/ami-amazon-linux-latest/amzn-ami-hvm-x86_64-ebs`。

有关在未指定实例时 AWS Data Pipeline 创建的默认 Amazon EC2 实例的信息，请参阅 Amazon Web Services Region 的默认 Amazon EC2 实例。

## 示例

### EC2-Classic

> **Important**
>
> 只有在 2013 年 12 月 4 日之前创建的 AWS 账户支持 EC2-Classic 平台。如果您拥有其中一个账户，则可以选择为 EC2-Classic 网络（而不是 VPC）中的管道创建 EC2Resource 对象。我们强烈建议为您在 VPC 中的所有管道创建资源。此外，如果您在 EC2-Classic 有现有资源，建议您把这些资源迁移到 VPC。

以下示例对象在 EC2-Classic 中启动 EC2 实例（带一些可选字段集）。

```
{
  "id" : "MyEC2Resource",
  "type" : "Ec2Resource",
  "actionOnTaskFailure" : "terminate",
  "actionOnResourceFailure" : "retryAll",
  "maximumRetries" : "1",
  "instanceType" : "m5.large",
  "securityGroups" : [
    "test-group",
    "default"
  ],
  "keyPair" : "my-key-pair"
}
```

### EC2-VPC

以下示例对象在非默认 VPC 中启动 EC2 实例 (设置了一些可选字段)。

```
{
  "id" : "MyEC2Resource",
  "type" : "Ec2Resource",
  "actionOnTaskFailure" : "terminate",
  "actionOnResourceFailure" : "retryAll",
  "maximumRetries" : "1",
```

```
  "instanceType" : "m5.large",
  "securityGroupIds" : [
    "sg-12345678",
    "sg-12345678"
  ],
  "subnetId": "subnet-12345678",
  "associatePublicIpAddress": "true",
  "keyPair" : "my-key-pair"
}
```

## 语法

| 必填字段 | 描述 | 槽类型 |
| --- | --- | --- |
| resourceRole | 控制 Amazon EC2 实例可访问的资源的 IAM 角色。 | String |
| role | AWS Data Pipeline 用于创建 EC2 实例的 IAM 角色。 | String |

| 对象调用字段 | 描述 | 槽类型 |
| --- | --- | --- |
| 计划 | 该对象在计划间隔的执行中调用。<br><br>要设置此对象的依赖项执行顺序，请指定对另一个对象的计划引用。您可以通过下列方式之一来执行该操作：<br><br>• 要确保管道中的所有对象都继承计划，请明确地在对象上设置计划：`"schedule": {"ref": "DefaultSchedule"}` 。在大多数情况下，将计划引用放在默认管道对象上会非常有用，这样所有对象都可以继承该计划。<br>• 如果管道有一些嵌套在主计划中的计划，则可以创建具有计划引用的父对象。有关示例可选计划配置的更多信息，请参阅 https:// | 引用对象，例如，`"schedule":{"ref": "myScheduleId"}` |

| 对象调用字段 | 描述 | 槽类型 |
| --- | --- | --- |
| | docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html。 | |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| actionOnResourceFailure | 在此资源发生资源失败后执行的操作。有效值为 `"retryall"` 和 `"retrynone"` 。 | String |
| actionOnTaskFailure | 在此资源发生任务失败后执行的操作。有效值为 `"continue"` 或 `"terminate"` 。 | String |
| associatePublicIpAddress | 指示是否向实例分配公有 IP 地址。如果实例位于 Amazon EC2 或 Amazon VPC 中，则默认值为`true`。否则，默认值为 `false`。 | 布尔值 |
| attemptStatus | 来自远程活动的最近报告的状态。 | String |
| attemptTimeout | 远程工作完成的超时时间。如果设置此字段，则可能会重试未在指定开始时间内完成的远程活动。 | 周期 |
| availabilityZone | 要在其中启动 Amazon EC2 实例的可用区。 | String |
| disableIMDSv1 | 默认值为 false，并启用 IMDSv1 和 IMDSv2。如果你将其设置为 true，那么它就会禁用 IMDSv1 并且只提供 IMDSv2 | 布尔值 |
| failureAndRerunMode | 描述依赖项失败或重新运行时的使用者节点行为。 | 枚举 |
| httpProxy | 客户端用来连接到 AWS 服务的代理主机。 | 引用对象，例如，`"httpProxy": {"ref":"myHttpProxyId"}` |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| imageId | 要用于实例的 AMI 的 ID。默认情况下，AWS Data Pipeline 使用 HVM AMI 虚拟化类型。使用的特定 AMI ID 取决于区域：您可以通过指定自己选择的 HVM AMI 来覆盖默认 AMI。有关 AMI 类型的更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的 Linux AMI 虚拟化类型和查找 Linux AMI。 | String |
| initTimeout | 资源启动前要等待的时间长度。 | 周期 |
| instanceCount | 已淘汰。 | 整数 |
| instanceType | 要启动的 Amazon EC2 实例的类型。 | String |
| keyPair | 密钥对的名称。如果您在未指定密钥对的情况下启动 Amazon EC2 实例，则无法登录该实例。 | String |
| lateAfterTimeout | 管道启动后经过的时间，在此时间内，对象必须完成。仅当计划类型未设置为 ondemand 时才会触发。 | 周期 |
| maxActiveInstances | 组件的并发活动实例的最大数量。重新运行不计入活动实例数中。 | 整数 |
| maximumRetries | 失败后的最大重试次数。 | 整数 |
| minInstanceCount | 已淘汰。 | 整数 |
| onFail | 当前对象失败时要运行的操作。 | 引用对象，例如， `"onFail": {"ref":"myActionId"}` |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| onLateAction | 在尚未计划对象或对象仍在运行的情况下将触发的操作。 | 引用对象，例如，`"onLateAction":{"ref":"myActionId"}` |
| onSuccess | 当前对象成功时要运行的操作。 | 引用对象，例如，`"onSuccess":{"ref":"myActionId"}` |
| parent | 作为槽继承源的当前对象的父项。 | 引用对象，例如， `"parent":{"ref":"myBaseObjectId"}` |
| pipelineLogUri | 用于上传管道日志的 Amazon S3 URI，例如 `'s3://BucketName/Key/'` 。 | String |
| region | 应在其中运行 Amazon EC2 实例的区域的代码。默认情况下，该实例在管道所在的区域中运行。您可以在从属数据集所在的区域中运行实例。 | 枚举 |
| reportProgressTimeout | 远程工作对 `reportProgress` 的连续调用的超时时间。如果设置此字段，则未报告指定时段的进度的远程活动可能会被视为停滞并且将进行重试。 | 周期 |
| retryDelay | 两次重试之间的超时时间。 | 周期 |
| runAsUser | 要运行 TaskRunner 的用户。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| runsOn | 禁止在该对象上使用此字段。 | 引用对象，例如，`"runsOn": {"ref":"myResourceId"}` |
| scheduleType | 您可以通过计划类型指定应在间隔开始时、间隔结束时还是按需计划管道定义中的对象。<br><br>值为：<br><br>• `timeseries` 在每个间隔结束时计划实例。<br>• `cron` 在每个间隔开始时计划实例。<br>• `ondemand` 可让您可以在每次激活时运行一次管道。您不需要克隆或重新创建管道以再次运行它。如果您使用按需计划，则必须在默认对象中指定它，并且必须是在管道中为对象指定的唯一 `scheduleType` 。要使用按需管道，请为每个后续运行调用 `ActivatePipeline` 操作。 | 枚举 |
| securityGroupIds | 要用于资源池中的实例的一个或多个 Amazon EC2 安全组的 ID。 | String |
| securityGroups | 要用于资源池中的实例的一个或多个 Amazon EC2 安全组。 | String |
| spotBidPrice | 每小时您的 Spot 实例的最高价 (美元)，是一个介于 0 和 20.00 (不含) 的小数值。 | String |
| subnetId | 要在其中启动实例的 Amazon EC2 子网的 ID。 | String |
| terminateAfter | 小时数，经过此时间后将终止资源。 | 周期 |
| useOnDemandOnLastAttempt | 在最后一次尝试请求 Spot 实例时，请求的是按需实例而不是 Spot 实例。这可确保如果所有之前的尝试都失败，则最后一次尝试不中断。 | 布尔值 |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| workerGroup | 禁止在该对象上使用此字段。 | String |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @activeInstances | 当前计划的有效实例对象的列表。 | 引用对象，例如，`"activeInstances": {"ref":"myRunnableObjectId"}` |
| @actualEndTime | 该对象的执行完成时间。 | DateTime |
| @actualStartTime | 该对象的执行开始时间。 | DateTime |
| cancellationReason | 该对象被取消时显示的 `cancellationReason` 。 | String |
| @cascadeFailedOn | 对象在其上失败的依赖项链的描述。 | 引用对象，例如，`"cascadeFailedOn": {"ref":"myRunnableObjectId"}` |
| emrStepLog | 仅在尝试 Amazon EMR 活动时可用的步骤日志。 | String |
| errorId | 该对象失败时显示的错误 ID。 | String |
| errorMessage | 该对象失败时显示的错误消息。 | String |
| errorStackTrace | 该对象失败时显示的错误堆栈跟踪。 | String |
| @failureReason | 资源失败的原因。 | String |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @finishedTime | 该对象完成其执行的时间。 | DateTime |
| hadoopJobLog | 在尝试 Amazon EMR 的活动时可用的 Hadoop 任务日志。 | String |
| @healthStatus | 对象的运行状况，反映进入终止状态的上个对象实例成功还是失败。 | String |
| @healthStatusFromInstanceId | 进入终止状态的上个实例对象的 ID。 | String |
| @healthStatusUpdatedTime | 上次更新运行状况的时间。 | DateTime |
| hostname | 已执行任务尝试的客户端的主机名。 | String |
| @lastDeactivatedTime | 上次停用该对象的时间。 | DateTime |
| @latestCompletedRunTime | 已完成执行的最新运行的时间。 | DateTime |
| @latestRunTime | 已计划执行的最新运行的时间。 | DateTime |
| @nextRunTime | 计划下次运行的时间。 | DateTime |
| reportProgressTime | 远程活动报告进度的最近时间。 | DateTime |
| @scheduledEndTime | 对象的计划结束时间。 | DateTime |
| @scheduledStartTime | 对象的计划开始时间。 | DateTime |
| @status | 该对象的状态。 | String |
| @version | 用来创建对象的管道版本。 | String |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @waitingOn | 此对象在其上处于等待状态的依赖项列表的描述。 | 引用对象，例如，`"waitingOn": {"ref":"myRunnableObjectId"}` |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息。 | String |
| @pipelineId | 该对象所属的管道的 ID。 | String |
| @sphere | 对象在生命周期中的位置。组件对象产生实例对象，后者执行尝试对象。 | String |

# EmrCluster

表示 Amazon EMR 群集的配置。EmrActivity 和 HadoopActivity 使用此对象来启动集群。

内容

- 调度器
- Amazon EMR 发行版
- Amazon EMR 权限
- 语法
- 示例
- 另请参阅

## 调度器

计划程序提供了一种方法来在 Hadoop 集群中指定资源分配和作业优先级。管理员或用户可以为各类用户和应用程序选择一个计划程序。计划程序可能使用队列来向用户和应用程序分配资源。您在创建集群时会设置这些队列。随后，您可以将特定类型的工作和用户设为优先于其他工作和用户。这样可以高效地使用集群资源，并允许多个用户将工作提交到集群。有以下三类计划程序可用：

- FairScheduler - 尝试在相当长的一段时间内均衡地计划资源。
- CapacityScheduler - 使用队列以允许集群管理员将用户分配给优先级和资源年分配各不相同的队列。
- Default - 由集群使用 (可由您的站点配置)。

## Amazon EMR 发行版

Amazon EMR 版本是一组来自大数据生态系统的开源应用程序。每个发行版由您在创建集群时选择让 Amazon EMR 安装和配置的各个大数据应用程序、组件和功能组成。可使用发行版标注指定版本。版本标签的格式是 `emr-x.x.x`。例如，`emr-5.30.0`。基于版本标签 `emr-4.0.0` 及更高版本的 Amazon EMR 集群使用 `releaseLabel` 属性指定 `EmrCluster` 对象的版本标签。早期版本使用 `amiVersion` 属性。

> **Important**
>
> 使用发布版本 5.22.0 或更高版本创建的所有 Amazon EMR 集群都使用签名版本 4 向 Amazon S3 验证请求。某些早期发布版本使用签名版本 2。对签名版本 2 的支持即将停止。有关更多信息，请参阅 Amazon S3 更新 — SIGv2 弃用期延长并修改。我们强烈建议您使用支持签名版本 4 的 Amazon EMR 发布版本。对于早期发布版本，从 EMR 4.7.x 开始，系列中的最新版本已更新为支持签名版本 4。使用较早版本的 EMR 时，建议您使用系列中的最新版本。此外，请避免早于 EMR 4.7.0 的版本。

### 注意事项和限制

#### 使用最新版本的任务运行程序

如果您将自管理的 `EmrCluster` 对象与版本标签结合使用，请使用最新的任务运行程序。有关任务运行程序的更多信息，请参阅使用任务运行程序。您可以为所有 Amazon EMR 配置分类配置属性值。有关更多信息，请参阅 Amazon EMR 版本指南中的配置应用程序以及 the section called “EmrConfiguration” 和 the section called “属性” 对象引用。

#### 支持 IMDSv2

此前，AWS Data Pipeline 仅支持 IMDSv1。目前，AWS Data Pipeline 在 Amazon EMR 5.23.1、5.27.1、5.32 或更高版本以及 Amazon EMR 6.2 或更高版本中支持 IMDSv2。在从实例检索元数据信息时，IMDSv2 使用面向会话的方法来更好地处理身份验证。您应该使用 TaskRunner-2.0 创建用户管理的资源，将您的实例配置为发出 IMDSv2 调用。

Amazon EMR 5.32 或更高版本以及 Amazon EMR 6.x

Amazon EMR 5.32 或更高版本和 6.x 版本系列使用 Hadoop 3.x 版本。Hadoop 3.x 版本与 Hadoop 2.x 版本相比，引入了对 Hadoop 类路径的评估方式的重大变更。像 Joda-Time 这样的常见库已从类路径中删除。

如果 EmrActivity 或 HadoopActivity 运行的 Jar 文件依赖于 Hadoop 3.x 中已删除的库，则该步骤将失败，并显示错误 `java.lang.NoClassDefFoundError` 或 `java.lang.ClassNotFoundException`。对于使用 Amazon EMR 5.x 发行版运行时不会出现问题的 Jar 文件，可能会发生这种情况。

要解决此问题，在启动 `EmrActivity` 或 `HadoopActivity` 之前，必须将 Jar 文件依赖关系复制到 `EmrCluster` 对象上的 Hadoop 类路径中。我们提供 bash 脚本来执行此操作。bash 脚本可在以下位置找到，例如 `us-west-2`，*`MyRegion`* 是您的 `EmrCluster` 对象运行的 AWS 区域。

```
s3://datapipeline-MyRegion/MyRegion/bootstrap-actions/latest/TaskRunner/copy-jars-to-hadoop-classpath.sh
```

脚本的运行方式取决于 `EmrActivity` 或 `HadoopActivity` 是在由 AWS Data Pipeline 托管的资源上运行还是由自管理的资源上运行。

如果您使用由 AWS Data Pipeline 托管的资源，请向 `EmrCluster` 对象添加 `bootstrapAction`。`bootstrapAction` 指定要复制作为参数的脚本和 Jar 文件。每个 `EmrCluster` 对象最多可以添加 255 个 `bootstrapAction` 字段，也可以向已有引导操作的 `EmrCluster` 对象添加 `bootstrapAction` 字段。

要将此脚本指定为引导操作，请使用以下语法，其中 `JarFileRegion` 是 Jar 文件的保存区域，每个 *`MyJarFilen`* 是 Amazon S3 中要复制到 Hadoop 类路径的 Jar 文件的绝对路径。请勿指定默认位于 Hadoop 类路径中的 Jar 文件。

```
s3://datapipeline-MyRegion/MyRegion/bootstrap-actions/latest/TaskRunner/copy-jars-to-hadoop-classpath.sh,JarFileRegion,MyJarFile1,MyJarFile2[, ...]
```

以下示例指定了一个引导操作，该操作将复制 Amazon S3 中的两个 Jar 文件：`my-jar-file.jar` 和 `emr-dynamodb-tool-4.14.0-jar-with-dependencies.jar`。此示例使用 us-west-2 区域。

```
{
  "id" : "MyEmrCluster",
```

```
  "type" : "EmrCluster",
  "keyPair" : "my-key-pair",
  "masterInstanceType" : "m5.xlarge",
  "coreInstanceType" : "m5.xlarge",
  "coreInstanceCount" : "2",
  "taskInstanceType" : "m5.xlarge",
  "taskInstanceCount": "2",
  "bootstrapAction" : ["s3://datapipeline-us-west-2/us-west-2/bootstrap-actions/
latest/TaskRunner/copy-jars-to-hadoop-classpath.sh,us-west-2,s3://path/to/my-jar-
file.jar,s3://dynamodb-dpl-us-west-2/emr-ddb-storage-handler/4.14.0/emr-dynamodb-
tools-4.14.0-jar-with-dependencies.jar"]
}
```

我们强烈建议保存并激活管道，以便对新 bootstrapAction 的更改生效。

如果您使用自管理资源，则可以将脚本下载到集群实例，然后使用 SSH 从命令行运行该脚本。该脚本将在该目录中创建一个名为 /etc/hadoop/conf/shellprofile.d 的目录和一个名为 datapipeline-jars.sh 的文件。作为命令行参数提供的 jar 文件被复制到脚本创建的名为 /home/hadoop/datapipeline_jars 的目录中。如果您的集群设置不同，请在下载脚本后对脚本进行相应的修改。

在命令行上运行脚本的语法与使用前一个示例中所示的 bootstrapAction 略有不同。参数之间应使用空格而不是逗号，如以下示例所示。

```
./copy-jars-to-hadoop-classpath.sh us-west-2 s3://path/to/my-jar-file.jar s3://
dynamodb-dpl-us-west-2/emr-ddb-storage-handler/4.14.0/emr-dynamodb-tools-4.14.0-jar-
with-dependencies.jar
```

## Amazon EMR 权限

当您创建自定义 IAM 角色时，请仔细考虑您的群集执行其工作所需的最小权限。请务必授予对所需资源的访问权，例如文件（在 Amazon S3 中）或数据（在 Amazon RDS、Amazon Redshift 或 DynamoDB 中）。如果您希望将 visibleToAllUsers 设置为 False，您的角色必须具有适当的权限才能执行此操作。请注意，DataPipelineDefaultRole 没有这些权限。您必须提供 DefaultDataPipelineResourceRole 和 DataPipelineDefaultRole 角色的联合作为 EmrCluster 对象角色或创建您自己的角色来实现此目的。

## 语法

| 对象调用字段 | 描述 | 槽类型 |
| --- | --- | --- |
| 计划 | 该对象在计划间隔的执行中调用。指定对另一个对象的计划引用，以便设置该对象的依赖项执行顺序。您可以明确设置针对该对象的计划以满足该要求，例如，指定 `"schedule": {"ref": "DefaultSchedule"}` 。在大多数情况下，最好将计划引用放在默认管道对象上，以便所有对象继承该计划。或者，如果管道具有一个计划树 (计划位于主计划中)，您可以创建具有计划引用的父对象。有关示例可选计划配置的更多信息，请参阅 https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html。 | 引用对象，例如， `"schedule":{"ref": "myScheduleId"}` |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| actionOnResourceFailure | 在此资源发生资源失败后执行的操作。有效值为“`retryall`”(在指定的持续时间内对集群重试所有任务) 和“`retrynone` ”。 | String |
| actionOnTaskFailure | 在此资源发生任务失败后执行的操作。有效值为“continue”(意味着不终止集群) 和“terminate”。 | String |
| additionalMasterSecurityGroupIds | EMR 集群的其他主安全组的标识符，其形式为 sg-01XXXX6a。有关更多信息，请参阅 Amazon EMR 管理指南中的 Amazon EMR 其他安全组。 | String |
| additionalSlaveSecurityGroupIds | EMR 集群的其他从属安全组的标识符，其形式为 `sg-01XXXX6a` 。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| amiVersion | Amazon EMR 用来安装群集节点的 Amazon Machine Image (AMI) 版本。有关更多信息，请参阅 Amazon EMR 管理指南。 | String |
| applications | 要安装在集群中的应用程序，带逗号分隔的参数。默认情况下，安装 Hive 和 Pig。该参数仅适用于 Amazon EMR 4.0 和更高版本。 | String |
| attemptStatus | 来自远程活动的最近报告的状态。 | String |
| attemptTimeout | 远程工作完成的超时时间。如果设置此字段，则可能会重试未在设定的开始时间内完成的远程活动。 | 周期 |
| availabilityZone | 用于运行集群的可用区。 | String |
| bootstrapAction | 在集群启动时要运行的操作。您可以指定逗号分隔的参数。要指定多个操作 (最多 255 个)，请添加多个 `bootstrapAction` 字段。默认行为是启动集群，而不执行任何引导操作。 | String |
| 配置 | Amazon EMR 群集的配置。该参数仅适用于 Amazon EMR 4.0 和更高版本。 | 引用对象，例如，`"configuration":{"ref":"myEmrConfigurationId"}` |
| coreInstanceBidPrice | 您愿意为 Amazon EC2 实例支付的最高 Spot 价格。如果指定了出价，Amazon EMR 将为实例组使用 Spot 实例。以 USD 为单位指定。 | String |
| coreInstanceCount | 要用于集群的核心节点的数目。 | 整数 |
| coreInstanceType | 要用于核心节点的 Amazon EC2 实例的类型。请参阅Amazon EMR 集群支持的 Amazon EC2 实例 。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| coreGroupConfiguration | Amazon EMR 集群核心实例组的配置。该参数仅适用于 Amazon EMR 4.0 和更高版本。 | 引用对象，例如，“configuration”: {“ref”: “myEmrConfigurationId”} |
| coreEbsConfiguration | 将附加到 Amazon EMR 集群的核心组中的每个核心节点的 Amazon EBS 卷的配置。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的支持 EBS 优化的实例类型。 | 引用对象，例如，“coreEbsConfiguration”: {“ref”: “myEbsConfiguration”} |
| customAmiId | 仅适用于 Amazon EMR 版本 5.7.0 及更高版本。指定当 Amazon EMR 预置 Amazon EC2 实例时要使用的自定义 AMI 的 AMI ID。也可以使用它来代替引导操作以自定义集群节点配置。有关更多信息，请参阅《Amazon EMR 管理指南》中的以下主题：使用自定义 AMI | String |
| EbsBlockDeviceConfig | 请求的与实例组关联的 Amazon EBS 块储存设备的配置。包含指定数量的卷，这些卷将与实例组中的每个实例相关联。包括 volumesPerInstance 和 volumeSpecification，其中：<br>• volumesPerInstance 是具有特定卷配置的 EBS 卷数，这些卷将与实例组中的每个实例相关联。<br>• volumeSpecification 是 Amazon EBS 卷规格，例如，为附加到 Amazon EMR 集群中的 EC2 实例的 EBS 卷请求的卷类型、IOPS 和大小 (GiB)。 | 引用对象，例如，“EbsBlockDeviceConfig”: {“ref”: “myEbsBlockDeviceConfig”} |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| emrManagedMasterSecurityGroupId | Amazon EMR 集群的主安全组的标识符，它采用 sg-01XXXX6a 格式。有关更多信息，请参阅 Amazon EMR 管理指南中的配置安全组。 | String |
| emrManagedSlaveSecurityGroupId | Amazon EMR 集群的从属安全组的标识符，它采用 sg-01XXXX6a 格式。 | String |
| enableDebugging | 在 Amazon EMR 集群上启用调试。 | String |
| failureAndRerunMode | 描述依赖项失败或重新运行时的使用者节点行为。 | 枚举 |
| hadoopSchedulerType | 集群的计划程序类型。有效类型为：PARALLEL_FAIR_SCHEDULING 、PARALLEL_CAPACITY_SCHEDULING 和 DEFAULT_SCHEDULER 。 | 枚举 |
| httpProxy | 客户端用来连接到 Amazon Web Services 的代理主机。 | 引用对象，例如，"httpProxy":{"ref":"myHttpProxyId"} |
| initTimeout | 资源启动前要等待的时间长度。 | 周期 |
| keyPair | 要用于登录 Amazon EMR 群集的主节点的 Amazon EC2 密钥对。 | String |
| lateAfterTimeout | 管道启动后经过的时间，在此时间内，对象必须完成。仅当计划类型未设置为 ondemand 时才会触发。 | 周期 |
| masterInstanceBidPrice | 您愿意为 Amazon EC2 实例支付的最高 Spot 价格。它是一个介于 0 和 20.00 之间（不含）的数字。以 USD 为单位指定。设置此值将为 Amazon EMR 集群主节点启用 Spot 实例。如果指定了出价，Amazon EMR 将为实例组使用 Spot 实例。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| masterInstanceType | 要用于主节点的 Amazon EC2 实例的类型。请参阅Amazon EMR 集群支持的 Amazon EC2 实例 。 | String |
| masterGroupConfiguration | Amazon EMR 集群主实例组的配置。该参数仅适用于 Amazon EMR 4.0 和更高版本。 | 引用对象，例如，“configuration”: {“ref”: “myEmrConfigurationId”} |
| masterEbsConfiguration | 将附加到 Amazon EMR 集群的主组中的每个主节点的 Amazon EBS 卷的配置。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的支持 EBS 优化的实例类型。 | 引用对象，例如，“masterEbsConfiguration”: {“ref”: “myEbsConfiguration”} |
| maxActiveInstances | 组件的并发活动实例的最大数量。重新运行不计入活动实例数中。 | 整数 |
| maximumRetries | 失败后的最大重试次数。 | 整数 |
| onFail | 当前对象失败时要运行的操作。 | 引用对象，例如，"onFail": {"ref":"myActionId"} |
| onLateAction | 在尚未计划对象或对象仍未完成的情况下将触发的操作。 | 引用对象，例如，"onLateAction":{"ref":"myActionId"} |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| onSuccess | 当前对象成功时要运行的操作。 | 引用对象，例如，`"onSuccess": {"ref":"myActionId"}` |
| parent | 作为槽继承源的当前对象的父项。 | 引用对象，例如，`"parent": {"ref":"myBaseObjectId"}` |
| pipelineLogUri | 用于上传管道日志的 Amazon S3 URI (例如“s3://BucketName/Key/”)。 | String |
| region | Amazon EMR 群集应在其中运行的区域的代码。默认情况下，该集群在管道所在的区域中运行。您可以在从属数据集所在的区域中运行集群。 | 枚举 |
| releaseLabel | EMR 集群的版本标签。 | String |
| reportProgressTimeout | 远程工作对 `reportProgress` 的连续调用的超时时间。如果设置此字段，则未报告指定时段的进度的远程活动可能会被视为停滞且已重试。 | 周期 |
| resourceRole | AWS Data Pipeline 用于创建 Amazon EMR 群集的 IAM 角色。默认角色是 `DataPipelineDefaultRole` 。 | String |
| retryDelay | 两次重试之间的超时时间。 | 周期 |
| role | 传递到 Amazon EMR 以创建 EC2 节点的 IAM 角色。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| runsOn | 禁止在该对象上使用此字段。 | 引用对象，例如，"runsOn": {"ref":"myResourceId"} |
| securityConfiguration | 应用于集群的 EMR 安全配置的标识符。该参数仅适用于 Amazon EMR 4.8.0 和更高版本。 | String |
| serviceAccessSecurityGroupId | Amazon EMR 集群的服务访问安全组的标识符。 | 字符串。它采用 sg-01XXXX6a 格式，例如，sg-1234abcd 。 |
| scheduleType | 您可以通过计划类型指定应在间隔开头还是结尾计划管道定义中的对象。值包括：cron、ondemand 和 timeseries 。timeseries 计划表示在每个间隔结尾计划实例。cron 计划表示在每个间隔开头计划实例。ondemand 计划让您可以在每次激活时运行一次管道。您不需要克隆或重新创建管道以再次运行它。如果您使用 ondemand 计划，则必须在默认对象中指定它，并且该计划必须是在管道中为对象指定的唯一 scheduleType 。要使用 ondemand 管道，请为每个后续运行调用 ActivatePipeline 操作。 | 枚举 |
| subnetId | 要在其中启动 Amazon EMR 集群的子网的标识符。 | String |
| supportedProducts | 在 Amazon EMR 集群上安装第三方软件的参数，例如，安装第三方 Hadoop 分发版本。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| taskInstanceBidPrice | 您愿意为 EC2 实例支付的最高 Spot 价格。一个介于 0 和 20.00 之间（不含）的数字。以 USD 为单位指定。如果指定了出价，Amazon EMR 将为实例组使用 Spot 实例。 | String |
| taskInstanceCount | 要用于 Amazon EMR 集群的任务节点数。 | 整数 |
| taskInstanceType | 要用于任务节点的 Amazon EC2 实例的类型。 | String |
| taskGroupConfiguration | Amazon EMR 集群任务实例组的配置。该参数仅适用于 Amazon EMR 4.0 和更高版本。 | 引用对象，例如，“configuration”: {“ref”: “myEmrConfigurationId”} |
| taskEbsConfiguration | 将附加到 Amazon EMR 集群的任务组中的每个任务节点的 Amazon EBS 卷的配置。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的支持 EBS 优化的实例类型。 | 引用对象，例如，“taskEbsConfiguration”: {“ref”: “myEbsConfiguration”} |
| terminateAfter | 终止资源之前经过的小时数。 | 整数 |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| VolumeSpecification | Amazon EBS 卷规格，例如，为附加到 Amazon EMR 集群中的 Amazon EC2 实例的 Amazon EBS 卷请求的卷类型、IOPS 和大小 (GiB)。节点可以是核心节点、主节点或任务节点。<br><br>VolumeSpecification 包括：<br><br>• iops() 整数。Amazon EBS 卷支持的每秒 I/O 操作数 (IOPS)，例如，1000。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的EBS I/O 特性。<br>• sizeinGB() 整数。Amazon EBS 卷大小 (GiB)，例如，500。有关有效的卷类型和硬盘驱动器大小组合的信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的 EBS 卷类型。<br>• volumetType 字符串。Amazon EBS 卷类型，例如，gp2。支持的卷类型包括标准、gp2、io1、st1、sc1 和其他。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的 EBS 卷类型。 | 引用对象，例如，“VolumeSpecification”: {“ref”: “myVolumeSpecification”} |
| useOnDemandOnLastAttempt | 在最后一次尝试请求资源时，请求的是按需实例而不是 Spot 实例。这可确保如果所有之前的尝试都失败，则最后一次尝试不中断。 | 布尔值 |
| workerGroup | 禁止在该对象中使用该字段。 | String |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @activeInstances | 当前计划的有效实例对象的列表。 | 引用对象，例如，"activeInstances":{ |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| | | "ref":"myRunnableObjectId"} |
| @actualEndTime | 该对象的执行完成时间。 | DateTime |
| @actualStartTime | 该对象的执行开始时间。 | DateTime |
| cancellationReason | 该对象被取消时显示的 cancellationReason。 | String |
| @cascadeFailedOn | 对象在其上失败的依赖项链的描述。 | 引用对象，例如，"cascadeFailedOn":{"ref":"myRunnableObjectId"} |
| emrStepLog | 仅在尝试 Amazon EMR 活动时可用的步骤日志。 | String |
| errorId | 该对象失败时显示的错误 ID。 | String |
| errorMessage | 该对象失败时显示的错误消息。 | String |
| errorStackTrace | 该对象失败时显示的错误堆栈跟踪。 | String |
| @failureReason | 资源失败的原因。 | String |
| @finishedTime | 该对象完成其执行的时间。 | DateTime |
| hadoopJobLog | 在尝试 Amazon EMR 的活动时可用的 Hadoop 任务日志。 | String |
| @healthStatus | 对象的运行状况，反映进入终止状态的上个对象实例成功还是失败。 | String |
| @healthStatusFromInstanceId | 进入终止状态的上个实例对象的 ID。 | String |
| @healthStatusUpdatedTime | 上次更新运行状况的时间。 | DateTime |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| hostname | 已执行任务尝试的客户端的主机名。 | String |
| @lastDeactivatedTime | 上次停用该对象的时间。 | DateTime |
| @latestCompletedRunTime | 已完成执行的最新运行的时间。 | DateTime |
| @latestRunTime | 已计划执行的最新运行的时间。 | DateTime |
| @nextRunTime | 计划下次运行的时间。 | DateTime |
| reportProgressTime | 远程活动报告进度的最近时间。 | DateTime |
| @scheduledEndTime | 对象的计划结束时间。 | DateTime |
| @scheduledStartTime | 对象的计划开始时间。 | DateTime |
| @status | 该对象的状态。 | String |
| @version | 用来创建对象的管道版本。 | String |
| @waitingOn | 此对象在其上处于等待状态的依赖项列表的描述。 | 引用对象，例如，"waitingOn":{"ref":"myRunnableObjectId"} |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息。 | String |
| @pipelineId | 该对象所属的管道的 ID。 | String |
| @sphere | 对象在生命周期中的位置。组件对象产生实例对象，后者执行尝试对象。 | String |

## 示例

下面是该对象类型的示例。

内容

- 使用 hadoopVersion 启动 Amazon EMR 集群
- 启动具有版本标签 emr-4.x 或更高版本的 Amazon EMR 集群
- 在您的 Amazon EMR 集群上安装额外的软件
- 在 3.x 版本上禁用服务器端加密
- 在 4.x 版本上禁用服务器端加密
- 在 HDFS 中配置 Hadoop KMS ACL 和创建加密区域
- 指定自定义 IAM 角色
- 使用适用于 Java 的 AWS 软件开发工具包中的 EmrCluster 资源
- 在私有子网中配置 Amazon EMR 集群
- 将 EBS 卷附加到集群节点

### 使用 hadoopVersion 启动 Amazon EMR 集群

Example

以下示例使用 AMI 1.0 版和 Hadoop 0.20 启动 Amazon EMR 群集。

```
{
  "id" : "MyEmrCluster",
  "type" : "EmrCluster",
  "hadoopVersion" : "0.20",
  "keyPair" : "my-key-pair",
  "masterInstanceType" : "m3.xlarge",
  "coreInstanceType" : "m3.xlarge",
  "coreInstanceCount" : "10",
  "taskInstanceType" : "m3.xlarge",
  "taskInstanceCount": "10",
  "bootstrapAction" : ["s3://Region.elasticmapreduce/bootstrap-actions/configure-
hadoop,arg1,arg2,arg3","s3://Region.elasticmapreduce/bootstrap-actions/configure-
hadoop/configure-other-stuff,arg1,arg2"]
}
```

启动具有版本标签 emr-4.x 或更高版本的 Amazon EMR 集群

Example

以下示例使用较新的 `releaseLabel` 字段启动 Amazon EMR 群集：

```
{
  "id" : "MyEmrCluster",
  "type" : "EmrCluster",
  "keyPair" : "my-key-pair",
  "masterInstanceType" : "m3.xlarge",
  "coreInstanceType" : "m3.xlarge",
  "coreInstanceCount" : "10",
  "taskInstanceType" : "m3.xlarge",
  "taskInstanceCount": "10",
  "releaseLabel": "emr-4.1.0",
  "applications": ["spark", "hive", "pig"],
  "configuration": {"ref":"myConfiguration"}
}
```

在您的 Amazon EMR 集群上安装额外的软件

Example

`EmrCluster` 提供了 `supportedProducts` 字段，它在 Amazon EMR 集群上安装第三方软件，例如，它用于安装 Hadoop 自定义分发版本（如 MapR）。它接受适用于第三方软件读取和处理的参数的逗号分隔列表。以下示例说明如何使用 `EmrCluster` 的 `supportedProducts` 字段来创建已安装 Karmasphere Analytics 的自定义 MapR M3 版本集群，并在该集群上运行 `EmrActivity` 对象。

```
{
    "id": "MyEmrActivity",
    "type": "EmrActivity",
    "schedule": {"ref": "ResourcePeriod"},
    "runsOn": {"ref": "MyEmrCluster"},
    "postStepCommand": "echo Ending job >> /mnt/var/log/stepCommand.txt",
    "preStepCommand": "echo Starting job > /mnt/var/log/stepCommand.txt",
    "step": "/home/hadoop/contrib/streaming/hadoop-streaming.jar,-input,s3n://
elasticmapreduce/samples/wordcount/input,-output, \
     hdfs:///output32113/,-mapper,s3n://elasticmapreduce/samples/wordcount/
wordSplitter.py,-reducer,aggregate"
  },
  {
    "id": "MyEmrCluster",
```

```
    "type": "EmrCluster",
    "schedule": {"ref": "ResourcePeriod"},
    "supportedProducts": ["mapr,--edition,m3,--version,1.2,--key1,value1","karmasphere-
enterprise-utility"],
    "masterInstanceType": "m3.xlarge",
    "taskInstanceType": "m3.xlarge"
}
```

在 3.x 版本上禁用服务器端加密

Example

默认情况下，AWS Data Pipeline 创建的 Hadoop 版本 2.x 的 EmrCluster 活动会启用服务器端加密。如果您想禁用服务器端加密，则必须在集群对象定义中指定引导操作。

以下示例创建一个已禁用服务器端加密的 EmrCluster 活动：

```
{
   "id":"NoSSEEmrCluster",
   "type":"EmrCluster",
   "hadoopVersion":"2.x",
   "keyPair":"my-key-pair",
   "masterInstanceType":"m3.xlarge",
   "coreInstanceType":"m3.large",
   "coreInstanceCount":"10",
   "taskInstanceType":"m3.large",
   "taskInstanceCount":"10",
   "bootstrapAction":["s3://Region.elasticmapreduce/bootstrap-actions/configure-
hadoop,-e, fs.s3.enableServerSideEncryption=false"]
}
```

在 4.x 版本上禁用服务器端加密

Example

您必须使用 EmrConfiguration 对象禁用服务器端加密。

以下示例创建一个已禁用服务器端加密的 EmrCluster 活动：

```
   {
      "name": "ReleaseLabelCluster",
      "releaseLabel": "emr-4.1.0",
      "applications": ["spark", "hive", "pig"],
      "id": "myResourceId",
```

```
      "type": "EmrCluster",
      "configuration": {
        "ref": "disableSSE"
      }
    },
    {
      "name": "disableSSE",
      "id": "disableSSE",
      "type": "EmrConfiguration",
      "classification": "emrfs-site",
      "property": [{
        "ref": "enableServerSideEncryption"
      }
      ]
    },
    {
      "name": "enableServerSideEncryption",
      "id": "enableServerSideEncryption",
      "type": "Property",
      "key": "fs.s3.enableServerSideEncryption",
      "value": "false"
    }
```

在 HDFS 中配置 Hadoop KMS ACL 和创建加密区域

Example

以下对象将为 Hadoop KMS 创建 ACL，并在 HDFS 中创建加密区域和相应的加密密钥：

```
{
      "name": "kmsAcls",
      "id": "kmsAcls",
      "type": "EmrConfiguration",
      "classification": "hadoop-kms-acls",
      "property": [
        {"ref":"kmsBlacklist"},
        {"ref":"kmsAcl"}
      ]
    },
    {
      "name": "hdfsEncryptionZone",
      "id": "hdfsEncryptionZone",
      "type": "EmrConfiguration",
      "classification": "hdfs-encryption-zones",
```

```
      "property": [
        {"ref":"hdfsPath1"},
        {"ref":"hdfsPath2"}
      ]
    },
    {
      "name": "kmsBlacklist",
      "id": "kmsBlacklist",
      "type": "Property",
      "key": "hadoop.kms.blacklist.CREATE",
      "value": "foo,myBannedUser"
    },
    {
      "name": "kmsAcl",
      "id": "kmsAcl",
      "type": "Property",
      "key": "hadoop.kms.acl.ROLLOVER",
      "value": "myAllowedUser"
    },
    {
      "name": "hdfsPath1",
      "id": "hdfsPath1",
      "type": "Property",
      "key": "/myHDFSPath1",
      "value": "path1_key"
    },
    {
      "name": "hdfsPath2",
      "id": "hdfsPath2",
      "type": "Property",
      "key": "/myHDFSPath2",
      "value": "path2_key"
    }
```

指定自定义 IAM 角色

Example

默认情况下，AWS Data Pipeline 将 `DataPipelineDefaultRole` 作为 Amazon EMR 服务角色传递，并将 `DataPipelineDefaultResourceRole` 作为 Amazon EC2 实例配置文件传递，以代表您创建资源。不过，您可以创建并使用自定义 Amazon EMR 服务角色和自定义实例配置文件。AWS Data Pipeline 应具有足够的权限来使用自定义角色创建集群，并且您必须将 AWS Data Pipeline 作为可信实体添加。

以下示例对象指定 Amazon EMR 集群的自定义角色：

```
{
   "id":"MyEmrCluster",
   "type":"EmrCluster",
   "hadoopVersion":"2.x",
   "keyPair":"my-key-pair",
   "masterInstanceType":"m3.xlarge",
   "coreInstanceType":"m3.large",
   "coreInstanceCount":"10",
   "taskInstanceType":"m3.large",
   "taskInstanceCount":"10",
   "role":"emrServiceRole",
   "resourceRole":"emrInstanceProfile"
}
```

使用适用于 Java 的 AWS 软件开发工具包中的 EmrCluster 资源

Example

以下示例说明了如何使用 EmrCluster 和 EmrActivity 创建 Amazon EMR 4.x 集群以通过 Java 软件开发工具包运行 Spark 步骤：

```
public class dataPipelineEmr4 {

  public static void main(String[] args) {

 AWSCredentials credentials = null;
 credentials = new ProfileCredentialsProvider("/path/to/
AwsCredentials.properties","default").getCredentials();
 DataPipelineClient dp = new DataPipelineClient(credentials);
 CreatePipelineRequest createPipeline = new
 CreatePipelineRequest().withName("EMR4SDK").withUniqueId("unique");
 CreatePipelineResult createPipelineResult = dp.createPipeline(createPipeline);
 String pipelineId = createPipelineResult.getPipelineId();

 PipelineObject emrCluster = new PipelineObject()
     .withName("EmrClusterObj")
     .withId("EmrClusterObj")
     .withFields(
   new Field().withKey("releaseLabel").withStringValue("emr-4.1.0"),
   new Field().withKey("coreInstanceCount").withStringValue("3"),
   new Field().withKey("applications").withStringValue("spark"),
```

```
    new Field().withKey("applications").withStringValue("Presto-Sandbox"),
    new Field().withKey("type").withStringValue("EmrCluster"),
    new Field().withKey("keyPair").withStringValue("myKeyName"),
    new Field().withKey("masterInstanceType").withStringValue("m3.xlarge"),
    new Field().withKey("coreInstanceType").withStringValue("m3.xlarge")
    );

 PipelineObject emrActivity = new PipelineObject()
     .withName("EmrActivityObj")
     .withId("EmrActivityObj")
     .withFields(
    new Field().withKey("step").withStringValue("command-runner.jar,spark-submit,--
executor-memory,1g,--class,org.apache.spark.examples.SparkPi,/usr/lib/spark/lib/spark-
examples.jar,10"),
    new Field().withKey("runsOn").withRefValue("EmrClusterObj"),
    new Field().withKey("type").withStringValue("EmrActivity")
    );

 PipelineObject schedule = new PipelineObject()
     .withName("Every 15 Minutes")
     .withId("DefaultSchedule")
     .withFields(
    new Field().withKey("type").withStringValue("Schedule"),
    new Field().withKey("period").withStringValue("15 Minutes"),
    new Field().withKey("startAt").withStringValue("FIRST_ACTIVATION_DATE_TIME")
    );

 PipelineObject defaultObject = new PipelineObject()
     .withName("Default")
     .withId("Default")
     .withFields(
    new Field().withKey("failureAndRerunMode").withStringValue("CASCADE"),
    new Field().withKey("schedule").withRefValue("DefaultSchedule"),
    new
 Field().withKey("resourceRole").withStringValue("DataPipelineDefaultResourceRole"),
    new Field().withKey("role").withStringValue("DataPipelineDefaultRole"),
    new Field().withKey("pipelineLogUri").withStringValue("s3://myLogUri"),
    new Field().withKey("scheduleType").withStringValue("cron")
    );

 List<PipelineObject> pipelineObjects = new ArrayList<PipelineObject>();

 pipelineObjects.add(emrActivity);
 pipelineObjects.add(emrCluster);
```

```
 pipelineObjects.add(defaultObject);
 pipelineObjects.add(schedule);

 PutPipelineDefinitionRequest putPipelineDefintion = new PutPipelineDefinitionRequest()
     .withPipelineId(pipelineId)
     .withPipelineObjects(pipelineObjects);

 PutPipelineDefinitionResult putPipelineResult =
 dp.putPipelineDefinition(putPipelineDefintion);
 System.out.println(putPipelineResult);

 ActivatePipelineRequest activatePipelineReq = new ActivatePipelineRequest()
     .withPipelineId(pipelineId);
 ActivatePipelineResult activatePipelineRes = dp.activatePipeline(activatePipelineReq);

      System.out.println(activatePipelineRes);
      System.out.println(pipelineId);

    }

}
```

在私有子网中配置 Amazon EMR 集群

Example

该示例包括在 VPC 的私有子网中启动集群的配置。有关更多信息，请参阅 Amazon EMR 管理指南中的在 VPC 中启动 Amazon EMR。此配置为可选配置。您可以在使用 EmrCluster 对象的任何管道中使用它。

要在私有子网中启动 Amazon EMR 集群，请在您的 SubnetId 配置中指定 emrManagedMasterSecurityGroupId、emrManagedSlaveSecurityGroupId、serviceAccessSec 和 EmrCluster。

```
{
  "objects": [
    {
      "output": {
        "ref": "S3BackupLocation"
      },
      "input": {
        "ref": "DDBSourceTable"
      },
```

```
      "maximumRetries": "2",
      "name": "TableBackupActivity",
      "step": "s3://dynamodb-emr-#{myDDBRegion}/emr-ddb-storage-handler/2.1.0/emr-
ddb-2.1.0.jar,org.apache.hadoop.dynamodb.tools.DynamoDbExport,#{output.directoryPath},#{input.t
      "id": "TableBackupActivity",
      "runsOn": {
        "ref": "EmrClusterForBackup"
      },
      "type": "EmrActivity",
      "resizeClusterBeforeRunning": "false"
    },
    {
      "readThroughputPercent": "#{myDDBReadThroughputRatio}",
      "name": "DDBSourceTable",
      "id": "DDBSourceTable",
      "type": "DynamoDBDataNode",
      "tableName": "#{myDDBTableName}"
    },
    {
      "directoryPath": "#{myOutputS3Loc}/#{format(@scheduledStartTime, 'YYYY-MM-dd-HH-
mm-ss')}",
      "name": "S3BackupLocation",
      "id": "S3BackupLocation",
      "type": "S3DataNode"
    },
    {
      "name": "EmrClusterForBackup",
      "coreInstanceCount": "1",
      "taskInstanceCount": "1",
      "taskInstanceType": "m4.xlarge",
      "coreInstanceType": "m4.xlarge",
      "releaseLabel": "emr-4.7.0",
      "masterInstanceType": "m4.xlarge",
      "id": "EmrClusterForBackup",
      "subnetId": "#{mySubnetId}",
      "emrManagedMasterSecurityGroupId": "#{myMasterSecurityGroup}",
      "emrManagedSlaveSecurityGroupId": "#{mySlaveSecurityGroup}",
      "serviceAccessSecurityGroupId": "#{myServiceAccessSecurityGroup}",
      "region": "#{myDDBRegion}",
      "type": "EmrCluster",
      "keyPair": "user-key-pair"
    },
    {
      "failureAndRerunMode": "CASCADE",
```

```
      "resourceRole": "DataPipelineDefaultResourceRole",
      "role": "DataPipelineDefaultRole",
      "pipelineLogUri": "#{myPipelineLogUri}",
      "scheduleType": "ONDEMAND",
      "name": "Default",
      "id": "Default"
    }
  ],
  "parameters": [
    {
      "description": "Output S3 folder",
      "id": "myOutputS3Loc",
      "type": "AWS::S3::ObjectKey"
    },
    {
      "description": "Source DynamoDB table name",
      "id": "myDDBTableName",
      "type": "String"
    },
    {
      "default": "0.25",
      "watermark": "Enter value between 0.1-1.0",
      "description": "DynamoDB read throughput ratio",
      "id": "myDDBReadThroughputRatio",
      "type": "Double"
    },
    {
      "default": "us-east-1",
      "watermark": "us-east-1",
      "description": "Region of the DynamoDB table",
      "id": "myDDBRegion",
      "type": "String"
    }
  ],
  "values": {
     "myDDBRegion": "us-east-1",
      "myDDBTableName": "ddb_table",
      "myDDBReadThroughputRatio": "0.25",
      "myOutputS3Loc": "s3://s3_path",
      "mySubnetId": "subnet_id",
      "myServiceAccessSecurityGroup":  "service access security group",
      "mySlaveSecurityGroup": "slave security group",
      "myMasterSecurityGroup": "master security group",
      "myPipelineLogUri": "s3://s3_path"
```

```
  }
}
```

将 EBS 卷附加到集群节点

Example

您可以将 EBS 卷附加到您的管道内的 EMR 集群中的任何类型的节点。要将 EBS 卷附加到节点，请在您的 `EmrCluster` 配置中使用 `coreEbsConfiguration`、`masterEbsConfiguration` 和 `TaskEbsConfiguration`。

该 Amazon EMR 集群示例使用 Amazon EBS 卷作为其主节点、任务节点和核心节点。有关更多信息，请参阅 Amazon EMR 管理指南中的 Amazon EMR 中的 Amazon EBS 卷。

这些配置是可选的。您可以在使用 `EmrCluster` 对象的任何管道中使用它们。

在管道中，单击 `EmrCluster` 对象配置，选择 Master EBS Configuration (主 EBS 配置)、Core EBS Configuration (核心 EBS 配置) 或 Task EBS Configuration (任务 EBS 配置)，然后输入类似于以下示例的配置详细信息。

```
{
  "objects": [
    {
      "output": {
        "ref": "S3BackupLocation"
      },
      "input": {
        "ref": "DDBSourceTable"
      },
      "maximumRetries": "2",
      "name": "TableBackupActivity",
      "step": "s3://dynamodb-emr-#{myDDBRegion}/emr-ddb-storage-handler/2.1.0/emr-
ddb-2.1.0.jar,org.apache.hadoop.dynamodb.tools.DynamoDbExport,#{output.directoryPath},#{input.t
      "id": "TableBackupActivity",
      "runsOn": {
        "ref": "EmrClusterForBackup"
      },
      "type": "EmrActivity",
      "resizeClusterBeforeRunning": "false"
    },
    {
      "readThroughputPercent": "#{myDDBReadThroughputRatio}",
```

```
      "name": "DDBSourceTable",
      "id": "DDBSourceTable",
      "type": "DynamoDBDataNode",
      "tableName": "#{myDDBTableName}"
    },
    {
      "directoryPath": "#{myOutputS3Loc}/#{format(@scheduledStartTime, 'YYYY-MM-dd-HH-
mm-ss')}",
      "name": "S3BackupLocation",
      "id": "S3BackupLocation",
      "type": "S3DataNode"
    },
    {
      "name": "EmrClusterForBackup",
      "coreInstanceCount": "1",
      "taskInstanceCount": "1",
      "taskInstanceType": "m4.xlarge",
      "coreInstanceType": "m4.xlarge",
      "releaseLabel": "emr-4.7.0",
      "masterInstanceType": "m4.xlarge",
      "id": "EmrClusterForBackup",
      "subnetId": "#{mySubnetId}",
      "emrManagedMasterSecurityGroupId": "#{myMasterSecurityGroup}",
      "emrManagedSlaveSecurityGroupId": "#{mySlaveSecurityGroup}",
      "region": "#{myDDBRegion}",
      "type": "EmrCluster",
      "coreEbsConfiguration": {
        "ref": "EBSConfiguration"
      },
      "masterEbsConfiguration": {
        "ref": "EBSConfiguration"
      },
      "taskEbsConfiguration": {
        "ref": "EBSConfiguration"
      },
      "keyPair": "user-key-pair"
    },
    {
       "name": "EBSConfiguration",
        "id": "EBSConfiguration",
        "ebsOptimized": "true",
        "ebsBlockDeviceConfig" : [
            { "ref": "EbsBlockDeviceConfig" }
        ],
```

```
        "type": "EbsConfiguration"
    },
    {
        "name": "EbsBlockDeviceConfig",
        "id": "EbsBlockDeviceConfig",
        "type": "EbsBlockDeviceConfig",
        "volumesPerInstance" : "2",
        "volumeSpecification" : {
            "ref": "VolumeSpecification"
        }
    },
    {
      "name": "VolumeSpecification",
      "id": "VolumeSpecification",
      "type": "VolumeSpecification",
      "sizeInGB": "500",
      "volumeType": "io1",
      "iops": "1000"
    },
    {
      "failureAndRerunMode": "CASCADE",
      "resourceRole": "DataPipelineDefaultResourceRole",
      "role": "DataPipelineDefaultRole",
      "pipelineLogUri": "#{myPipelineLogUri}",
      "scheduleType": "ONDEMAND",
      "name": "Default",
      "id": "Default"
    }
  ],
  "parameters": [
    {
      "description": "Output S3 folder",
      "id": "myOutputS3Loc",
      "type": "AWS::S3::ObjectKey"
    },
    {
      "description": "Source DynamoDB table name",
      "id": "myDDBTableName",
      "type": "String"
    },
    {
      "default": "0.25",
      "watermark": "Enter value between 0.1-1.0",
      "description": "DynamoDB read throughput ratio",
```

```
      "id": "myDDBReadThroughputRatio",
      "type": "Double"
    },
    {
      "default": "us-east-1",
      "watermark": "us-east-1",
      "description": "Region of the DynamoDB table",
      "id": "myDDBRegion",
      "type": "String"
    }
  ],
  "values": {
     "myDDBRegion": "us-east-1",
      "myDDBTableName": "ddb_table",
      "myDDBReadThroughputRatio": "0.25",
      "myOutputS3Loc": "s3://s3_path",
      "mySubnetId": "subnet_id",
      "mySlaveSecurityGroup": "slave security group",
      "myMasterSecurityGroup": "master security group",
      "myPipelineLogUri": "s3://s3_path"
  }
}
```

### 另请参阅

- EmrActivity

## HttpProxy

HttpProxy 使您能够配置自己的代理并让任务运行程序通过此代理访问 AWS Data Pipeline 服务。您不需要使用此信息配置正在运行的任务运行程序。

### TaskRunner 中的 HttpProxy 示例

以下管道定义显示一个 HttpProxy 对象：

```
{
  "objects": [
    {
      "schedule": {
        "ref": "Once"
      },
```

```
      "pipelineLogUri": "s3://myDPLogUri/path",
      "name": "Default",
      "id": "Default"
    },
    {
      "name": "test_proxy",
      "hostname": "hostname",
      "port": "port",
      "username": "username",
      "*password": "password",
      "windowsDomain": "windowsDomain",
      "type": "HttpProxy",
      "id": "test_proxy",
    },
    {
      "name": "ShellCommand",
      "id": "ShellCommand",
      "runsOn": {
        "ref": "Resource"
      },
      "type": "ShellCommandActivity",
      "command": "echo 'hello world' "
    },
    {
      "period": "1 day",
      "startDateTime": "2013-03-09T00:00:00",
      "name": "Once",
      "id": "Once",
      "endDateTime": "2013-03-10T00:00:00",
      "type": "Schedule"
    },
    {
      "role": "dataPipelineRole",
      "httpProxy": {
        "ref": "test_proxy"
      },
      "actionOnResourceFailure": "retrynone",
      "maximumRetries": "0",
      "type": "Ec2Resource",
      "terminateAfter": "10 minutes",
      "resourceRole": "resourceRole",
      "name": "Resource",
      "actionOnTaskFailure": "terminate",
      "securityGroups": "securityGroups",
```

```
      "keyPair": "keyPair",
      "id": "Resource",
      "region": "us-east-1"
    }
  ],
  "parameters": []
}
```

## 语法

| 必填字段 | 描述 | 槽类型 |
| --- | --- | --- |
| hostname | 客户端将用来连接到 Amazon Web Services 的代理的主机。 | String |
| port | 客户端将用来连接到 Amazon Web Services 的代理主机的端口。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |
| *password | 代理的密码。 | String |
| s3NoProxy | 在连接到 Amazon S3 时禁用 HTTP 代理 | 布尔值 |
| username | 代理的用户名。 | String |
| windowsDomain | NTLM 代理的 Windows 域名。 | String |
| windowsWorkgroup | NTLM 代理的 Windows 工作组名。 | String |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @version | 用来创建对象的管道版本。 | String |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息。 | String |
| @pipelineId | 该对象所属的管道的 ID。 | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象。 | String |

# 先决条件

以下是 AWS Data Pipeline 先决条件对象：

Objects

- DynamoDBDataExists
- DynamoDBTableExists
- 存在
- S3KeyExists
- S3PrefixNotEmpty
- ShellCommandPrecondition

## DynamoDBDataExists

一个用于检查 DynamoDB 表中是否有数据的先决条件。

### 语法

| 必填字段 | 描述 | 槽类型 |
| --- | --- | --- |
| role | 指定用于执行先决条件的角色。 | String |

| 必填字段 | 描述 | 槽类型 |
| --- | --- | --- |
| tableName | 要检查的 DynamoDB 表。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| attemptStatus | 来自远程活动的最近报告的状态。 | String |
| attemptTimeout | 远程工作完成的超时时间。如果设置此字段，则可能会重试未在设定的开始时间内完成的远程活动。 | 周期 |
| failureAndRerunMode | 描述依赖项失败或重新运行时的使用者节点行为。 | 枚举 |
| lateAfterTimeout | 管道启动后经过的时间，在此时间内，对象必须完成。仅当计划类型未设置为 ondemand 时才会触发。 | 周期 |
| maximumRetries | 失败后的最大重试次数 | 整数 |
| onFail | 当前对象失败时要运行的操作。 | 引用对象，例如，"onFail":{"ref":"myActionId"} |
| onLateAction | 在尚未计划对象或对象仍未完成的情况下将触发的操作。 | 引用对象，例如，"onLateAction":{"ref":"myActionId"} |
| onSuccess | 当前对象成功时要运行的操作。 | 引用对象，例如，"onSuccess":{"ref":"myActionId"} |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| preconditionTimeout | 从开始算起的时段，在该时段后，如果仍未满足先决条件，则会将先决条件标记为失败。 | 周期 |
| reportProgressTimeout | 远程工作对 reportProgress 的连续调用的超时时间。如果设置此字段，则未报告指定时段的进度的远程活动可能会被视为停滞且已重试。 | 周期 |
| retryDelay | 两次重试之间的超时时间。 | 周期 |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @activeInstances | 当前计划的有效实例对象的列表。 | 引用对象，例如，"activeInstances":{"ref":"myRunnableObjectId"} |
| @actualEndTime | 该对象的执行完成时间。 | DateTime |
| @actualStartTime | 该对象的执行开始时间。 | DateTime |
| cancellationReason | 该对象被取消时显示的 cancellationReason。 | String |
| @cascadeFailedOn | 对象在其上失败的依赖项链的描述。 | 引用对象，例如，"cascadeFailedOn":{"ref":"myRunnableObjectId"} |
| currentRetryCount | 在此尝试中已经重试先决条件的次数。 | String |
| emrStepLog | 仅在尝试 EMR 活动时可用的 EMR 步骤日志 | String |
| errorId | 该对象失败时显示的 errorId。 | String |
| errorMessage | 该对象失败时显示的 errorMessage。 | String |
| errorStackTrace | 该对象失败时显示的错误堆栈跟踪。 | String |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| hadoopJobLog | 在尝试基于 EMR 的活动时可用的 Hadoop 任务日志。 | String |
| hostname | 已执行任务尝试的客户端的主机名。 | String |
| lastRetryTime | 在此尝试中上次重试先决条件的时间。 | String |
| node | 将为其执行此先决条件的节点。 | 引用对象，例如，"node":{"ref":"myRunnableObjectId"} |
| reportProgressTime | 远程活动报告进度的最近时间。 | DateTime |
| @scheduledEndTime | 对象的计划结束时间。 | DateTime |
| @scheduledStartTime | 对象的计划开始时间。 | DateTime |
| @status | 该对象的状态。 | String |
| @version | 用来创建对象的管道版本。 | String |
| @waitingOn | 该对象在其上处于等待状态的依赖项列表的描述。 | 引用对象，例如，"waitingOn":{"ref":"myRunnableObjectId"} |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息。 | String |
| @pipelineId | 该对象所属的管道的 ID。 | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象。 | String |

# DynamoDBTableExists

一个用于检查 DynamoDB 表是否存在的先决条件。

## 语法

| 必填字段 | 描述 | 槽类型 |
| --- | --- | --- |
| role | 指定用于执行先决条件的角色。 | String |
| tableName | 要检查的 DynamoDB 表。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| attemptStatus | 来自远程活动的最近报告的状态。 | String |
| attemptTimeout | 远程工作完成的超时时间。如果设置此字段，则可能会重试未在设定的开始时间内完成的远程活动。 | 周期 |
| failureAndRerunMode | 描述依赖项失败或重新运行时的使用者节点行为。 | 枚举 |
| lateAfterTimeout | 管道启动后经过的时间，在此时间内，对象必须完成。仅当计划类型未设置为 ondemand 时才会触发。 | 周期 |
| maximumRetries | 失败后的最大重试次数 | 整数 |
| onFail | 当前对象失败时要运行的操作。 | 引用对象，例如，"onFail":{"ref":"myActionId"} |
| onLateAction | 在尚未计划对象或对象仍未完成的情况下将触发的操作。 | 引用对象，例如，"onLateAction":{"ref":"myActionId"} |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| onSuccess | 当前对象成功时要运行的操作。 | 引用对象，例如，"onSuccess":{"ref":"myActionId"} |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |
| preconditionTimeout | 从开始算起的时段，在该时段后，如果仍未满足先决条件，则会将先决条件标记为失败。 | 周期 |
| reportProgressTimeout | 远程工作对 reportProgress 的连续调用的超时时间。如果设置此字段，则未报告指定时段的进度的远程活动可能会被视为停滞且已重试。 | 周期 |
| retryDelay | 两次重试之间的超时时间。 | 周期 |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @activeInstances | 当前计划的有效实例对象的列表。 | 引用对象，例如，"activeInstances":{"ref":"myRunnableObjectId"} |
| @actualEndTime | 该对象的执行完成时间。 | DateTime |
| @actualStartTime | 该对象的执行开始时间。 | DateTime |
| cancellationReason | 该对象被取消时显示的 cancellationReason。 | String |
| @cascadeFailedOn | 对象在其上失败的依赖项链的描述。 | 引用对象，例如，"cascadeFailedOn":{"ref":"myRunnableObjectId"} |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| currentRetryCount | 在此尝试中已经重试先决条件的次数。 | String |
| emrStepLog | 仅在尝试 EMR 活动时可用的 EMR 步骤日志 | String |
| errorId | 该对象失败时显示的 errorId。 | String |
| errorMessage | 该对象失败时显示的 errorMessage。 | String |
| errorStackTrace | 该对象失败时显示的错误堆栈跟踪。 | String |
| hadoopJobLog | 在尝试基于 EMR 的活动时可用的 Hadoop 任务日志。 | String |
| hostname | 已执行任务尝试的客户端的主机名。 | String |
| lastRetryTime | 在此尝试中上次重试先决条件的时间。 | String |
| node | 将为其执行此先决条件的节点。 | 引用对象，例如，"node":{"ref":"myRunnableObjectId"} |
| reportProgressTime | 远程活动报告进度的最近时间。 | DateTime |
| @scheduledEndTime | 对象的计划结束时间。 | DateTime |
| @scheduledStartTime | 对象的计划开始时间。 | DateTime |
| @status | 该对象的状态。 | String |
| @version | 用来创建对象的管道版本。 | String |
| @waitingOn | 该对象在其上处于等待状态的依赖项列表的描述。 | 引用对象，例如，"waitingOn":{"ref":"myRunnableObjectId"} |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息 | String |
| @pipelineId | 该对象所属的管道的 ID | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象 | String |

## 存在

检查数据节点对象是否存在。

> **Note**
>
> 我们建议您改用系统管理的先决条件。有关更多信息，请参阅先决条件。

### 示例

以下是该对象类型的示例。InputData 对象引用该对象（即 Ready）以及您在同一管道定义文件中定义的另一个对象。CopyPeriod 为 Schedule 对象。

```
{
  "id" : "InputData",
  "type" : "S3DataNode",
  "schedule" : { "ref" : "CopyPeriod" },
  "filePath" : "s3://example-bucket/InputData/#{@scheduledStartTime.format('YYYY-MM-dd-hh:mm')}.csv",
  "precondition" : { "ref" : "Ready" }
},
{
  "id" : "Ready",
  "type" : "Exists"
}
```

## 语法

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| attemptStatus | 来自远程活动的最近报告的状态。 | String |
| attemptTimeout | 远程工作完成的超时时间。如果设置此字段，则可能会重试未在设定的开始时间内完成的远程活动。 | 周期 |
| failureAndRerunMode | 描述依赖项失败或重新运行时的使用者节点行为。 | 枚举 |
| lateAfterTimeout | 管道启动后经过的时间，在此时间内，对象必须完成。仅当计划类型未设置为 ondemand 时才会触发。 | 周期 |
| maximumRetries | 失败后的最大重试次数 | 整数 |
| onFail | 当前对象失败时要运行的操作。 | 引用对象，例如，"onFail":{"ref":"myActionId"} |
| onLateAction | 在尚未计划对象或对象仍未完成的情况下将触发的操作。 | 引用对象，例如，"onLateAction":{"ref":"myActionId"} |
| onSuccess | 当前对象成功时要运行的操作。 | 引用对象，例如，"onSuccess":{"ref":"myActionId"} |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |
| preconditionTimeout | 从开始算起的时段，在该时段后，如果仍未满足先决条件，则会将先决条件标记为失败。 | 周期 |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| reportProgressTimeout | 远程工作对 reportProgress 的连续调用的超时时间。如果设置此字段，则未报告指定时段的进度的远程活动可能会被视为停滞且已重试。 | 周期 |
| retryDelay | 两次重试之间的超时时间。 | 周期 |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @activeInstances | 当前计划的有效实例对象的列表。 | 引用对象，例如，"activeInstances":{"ref":"myRunnableObjectId"} |
| @actualEndTime | 该对象的执行完成时间。 | DateTime |
| @actualStartTime | 该对象的执行开始时间。 | DateTime |
| cancellationReason | 该对象被取消时显示的 cancellationReason。 | String |
| @cascadeFailedOn | 对象在其上失败的依赖项链的描述。 | 引用对象，例如，"cascadeFailedOn":{"ref":"myRunnableObjectId"} |
| emrStepLog | 仅在尝试 EMR 活动时可用的 EMR 步骤日志 | String |
| errorId | 该对象失败时显示的 errorId。 | String |
| errorMessage | 该对象失败时显示的 errorMessage。 | String |
| errorStackTrace | 该对象失败时显示的错误堆栈跟踪。 | String |
| hadoopJobLog | 在尝试基于 EMR 的活动时可用的 Hadoop 任务日志。 | String |
| hostname | 已执行任务尝试的客户端的主机名。 | String |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| node | 将为其执行此先决条件的节点。 | 引用对象，例如，"node":{"ref":"myRunnableObjectId"} |
| reportProgressTime | 远程活动报告进度的最近时间。 | DateTime |
| @scheduledEndTime | 对象的计划结束时间。 | DateTime |
| @scheduledStartTime | 对象的计划开始时间。 | DateTime |
| @status | 该对象的状态。 | String |
| @version | 用来创建对象的管道版本。 | String |
| @waitingOn | 该对象在其上处于等待状态的依赖项列表的描述。 | 引用对象，例如，"waitingOn":{"ref":"myRunnableObjectId"} |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息。 | String |
| @pipelineId | 该对象所属的管道的 ID。 | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象。 | String |

### 另请参阅

- ShellCommandPrecondition

## S3KeyExists

检查 Amazon S3 数据节点中是否存在键。

## 示例

以下是该对象类型的示例。如果 s3Key 参数引用的密钥 s3://mybucket/mykey 已存在，则触发先决条件。

```
{
"id" : "InputReady",
"type" : "S3KeyExists",
"role" : "test-role",
"s3Key" : "s3://mybucket/mykey"
}
```

您还可以使用 S3KeyExists 作为第二个管道的先决条件，该先决条件等待第一个管道完成。为此，请执行以下操作：

1. 当第一个管道完成后，将一个文件写入到 Amazon S3。
2. 在第二个管道上创建 S3KeyExists 先决条件。

## 语法

| 必填字段 | 描述 | 槽类型 |
| --- | --- | --- |
| role | 指定用于执行先决条件的角色。 | String |
| s3Key | Amazon S3 密钥。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| attemptStatus | 来自远程活动的最近报告的状态。 | String |
| attemptTimeout | 再次尝试完成远程工作之前的超时时间。如果设置此字段，则可能会再次尝试未在启动后的设定时间内完成的远程活动。 | 周期 |
| failureAndRerunMode | 描述依赖项失败或重新运行时的使用者节点行为。 | 枚举 |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| lateAfterTimeout | 管道启动后经过的时间，在此时间内，对象必须完成。仅当计划类型未设置为 ondemand 时才会触发。 | 周期 |
| maximumRetries | 失败时启动尝试的最大次数。 | 整数 |
| onFail | 当前对象失败时要运行的操作。 | 引用对象，例如，"onFail":{"ref":"myActionId"} |
| onLateAction | 在尚未计划对象或对象仍未完成的情况下将触发的操作。 | 引用对象，例如，"onLateAction":{"ref":"myActionId"} |
| onSuccess | 当前对象成功时要运行的操作。 | 引用对象，例如，"onSuccess":{"ref":"myActionId"} |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |
| preconditionTimeout | 从开始算起的时段，在该时段后，如果仍未满足先决条件，则会将先决条件标记为失败。 | 周期 |
| reportProgressTimeout | 远程工作对 `reportProgress` 的连续调用的超时时间。如果设置此字段，则未报告指定时段的进度的远程活动可能会被视为停滞且已重试。 | 周期 |
| retryDelay | 两次连续尝试之间的超时时间。 | 周期 |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @activeInstances | 当前计划的有效实例对象的列表。 | 引用对象，例如，"activeInstances":{ |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| | | "ref":"myRunnableObjectId"} |
| @actualEndTime | 该对象的执行完成时间。 | DateTime |
| @actualStartTime | 该对象的执行开始时间。 | DateTime |
| cancellationReason | 该对象被取消时显示的 cancellationReason。 | String |
| @cascadeFailedOn | 对象在其上失败的依赖项链的描述。 | 引用对象，例如，"cascadeFailedOn":{"ref":"myRunnableObjectId"} |
| currentRetryCount | 在此尝试中已经重试先决条件的次数。 | String |
| emrStepLog | 仅在尝试 EMR 活动时可用的 EMR 步骤日志 | String |
| errorId | 该对象失败时显示的 errorId。 | String |
| errorMessage | 该对象失败时显示的 errorMessage。 | String |
| errorStackTrace | 该对象失败时显示的错误堆栈跟踪。 | String |
| hadoopJobLog | 在尝试基于 EMR 的活动时可用的 Hadoop 任务日志。 | String |
| hostname | 已执行任务尝试的客户端的主机名。 | String |
| lastRetryTime | 在此尝试中上次重试先决条件的时间。 | String |
| node | 将为其执行此先决条件的节点。 | 引用对象，例如，"node":{"ref":"myRunnableObjectId"} |
| reportProgressTime | 远程活动报告进度的最近时间。 | DateTime |
| @scheduledEndTime | 对象的计划结束时间。 | DateTime |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @scheduledStartTime | 对象的计划开始时间。 | DateTime |
| @status | 该对象的状态。 | String |
| @version | 用来创建对象的管道版本。 | String |
| @waitingOn | 该对象在其上处于等待状态的依赖项列表的描述。 | 引用对象，例如，"waitingOn":{"ref": "myRunnableObjectId"} |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息 | String |
| @pipelineId | 该对象所属的管道的 ID | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象 | String |

### 另请参阅

- ShellCommandPrecondition

## S3PrefixNotEmpty

一个用于检查是否存在带给定前缀（表示为 URI）的 Amazon S3 对象的先决条件。

### 示例

以下是使用必填字段、可选字段和表达式字段的该对象类型的示例。

```
{
  "id" : "InputReady",
```

```
  "type" : "S3PrefixNotEmpty",
  "role" : "test-role",
  "s3Prefix" : "#{node.filePath}"
}
```

## 语法

| 必填字段 | 描述 | 槽类型 |
| --- | --- | --- |
| role | 指定用于执行先决条件的角色。 | String |
| s3Prefix | 用于检查对象是否存在的 Amazon S3 前缀。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| attemptStatus | 来自远程活动的最近报告的状态。 | String |
| attemptTimeout | 远程工作完成的超时时间。如果设置此字段，则可能会重试未在设定的开始时间内完成的远程活动。 | 周期 |
| failureAndRerunMode | 描述依赖项失败或重新运行时的使用者节点行为。 | 枚举 |
| lateAfterTimeout | 管道启动后经过的时间，在此时间内，对象必须完成。仅当计划类型未设置为 ondemand 时才会触发。 | 周期 |
| maximumRetries | 失败后的最大重试次数 | 整数 |
| onFail | 当前对象失败时要运行的操作。 | 引用对象，例如，"onFail":{"ref":"myActionId"} |
| onLateAction | 在尚未计划对象或对象仍未完成的情况下将触发的操作。 | 引用对象，例如，"onLateAction":{"ref":"myActionId"} |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| onSuccess | 当前对象成功时要运行的操作。 | 引用对象，例如，"onSuccess":{"ref":"myActionId"} |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |
| preconditionTimeout | 从开始算起的时段，在该时段后，如果仍未满足先决条件，则会将先决条件标记为失败。 | 周期 |
| reportProgressTimeout | 远程工作对 reportProgress 的连续调用的超时时间。如果设置此字段，则未报告指定时段的进度的远程活动可能会被视为停滞且已重试。 | 周期 |
| retryDelay | 两次重试之间的超时时间。 | 周期 |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @activeInstances | 当前计划的有效实例对象的列表。 | 引用对象，例如，"activeInstances":{"ref":"myRunnableObjectId"} |
| @actualEndTime | 该对象的执行完成时间。 | DateTime |
| @actualStartTime | 该对象的执行开始时间。 | DateTime |
| cancellationReason | 该对象被取消时显示的 cancellationReason。 | String |
| @cascadeFailedOn | 对象在其上失败的依赖项链的描述。 | 引用对象，例如，"cascadeFailedOn":{"ref":"myRunnableObjectId"} |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| currentRetryCount | 在此尝试中已经重试先决条件的次数。 | String |
| emrStepLog | 仅在尝试 EMR 活动时可用的 EMR 步骤日志 | String |
| errorId | 该对象失败时显示的 errorId。 | String |
| errorMessage | 该对象失败时显示的 errorMessage。 | String |
| errorStackTrace | 该对象失败时显示的错误堆栈跟踪。 | String |
| hadoopJobLog | 在尝试基于 EMR 的活动时可用的 Hadoop 任务日志。 | String |
| hostname | 已执行任务尝试的客户端的主机名。 | String |
| lastRetryTime | 在此尝试中上次重试先决条件的时间。 | String |
| node | 将为其执行此先决条件的节点。 | 引用对象，例如，"node":{"ref":"myRunnableObjectId"} |
| reportProgressTime | 远程活动报告进度的最近时间。 | DateTime |
| @scheduledEndTime | 对象的计划结束时间。 | DateTime |
| @scheduledStartTime | 对象的计划开始时间。 | DateTime |
| @status | 该对象的状态。 | String |
| @version | 用来创建对象的管道版本。 | String |
| @waitingOn | 该对象在其上处于等待状态的依赖项列表的描述。 | 引用对象，例如，"waitingOn":{"ref":"myRunnableObjectId"} |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息 | String |
| @pipelineId | 该对象所属的管道的 ID | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象 | String |

### 另请参阅

- ShellCommandPrecondition

## ShellCommandPrecondition

一条可作为先决条件运行的 Unix/Linux shell 命令。

### 示例

以下是该对象类型的示例。

```
{
  "id" : "VerifyDataReadiness",
  "type" : "ShellCommandPrecondition",
  "command" : "perl check-data-ready.pl"
}
```

### 语法

| 所需的组 (下列选项之一是必需的) | 描述 | 槽类型 |
| --- | --- | --- |
| 命令 | 要运行的命令。此值与任何关联参数必须在从中运行任务运行程序的环境中起作用。 | String |
| scriptUri | 要下载并作为 shell 命令运行的文件的 Amazon S3 URI 路径。只应出现一个 scriptUri 或命令字段。scriptUri 无法使用参数，请使用命令。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| attemptStatus | 来自远程活动的最近报告的状态。 | String |
| attemptTimeout | 远程工作完成的超时时间。如果设置此字段，则可能会重试未在设定的开始时间内完成的远程活动。 | 周期 |
| failureAndRerunMode | 描述依赖项失败或重新运行时的使用者节点行为。 | 枚举 |
| lateAfterTimeout | 管道启动后经过的时间，在此时间内，对象必须完成。仅当计划类型未设置为 ondemand 时才会触发。 | 周期 |
| maximumRetries | 失败后的最大重试次数 | 整数 |
| onFail | 当前对象失败时要运行的操作。 | 引用对象，例如，"onFail":{"ref":"myActionId"} |
| onLateAction | 在尚未计划对象或对象仍未完成的情况下将触发的操作。 | 引用对象，例如，"onLateAction":{"ref":"myActionId"} |
| onSuccess | 当前对象成功时要运行的操作。 | 引用对象，例如，"onSuccess":{"ref":"myActionId"} |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |
| preconditionTimeout | 从开始算起的时段，在该时段后，如果仍未满足先决条件，则会将先决条件标记为失败。 | 周期 |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| reportProgressTimeout | 远程工作对 reportProgress 的连续调用的超时时间。如果设置此字段，则未报告指定时段的进度的远程活动可能会被视为停滞且已重试。 | 周期 |
| retryDelay | 两次重试之间的超时时间。 | 周期 |
| scriptArgument | 要传递到 shell 脚本的参数。 | String |
| stderr | 接收来自命令的重定向系统错误消息的 Amazon S3 路径。如果您使用 `runsOn` 字段，则由于运行活动的资源的短期性质，该字段必须为 Amazon S3 路径。不过，如果指定 `workerGroup` 字段，则允许使用本地文件路径。 | String |
| stdout | 接收来自命令的重定向输出的 Amazon S3 路径。如果您使用 `runsOn` 字段，则由于运行活动的资源的短期性质，该字段必须为 Amazon S3 路径。不过，如果指定 `workerGroup` 字段，则允许使用本地文件路径。 | String |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @activeInstances | 当前计划的有效实例对象的列表。 | 引用对象，例如，"activeInstances":{"ref":"myRunnableObjectId"} |
| @actualEndTime | 该对象的执行完成时间。 | DateTime |
| @actualStartTime | 该对象的执行开始时间。 | DateTime |
| cancellationReason | 该对象被取消时显示的 cancellationReason。 | String |
| @cascadeFailedOn | 对象在其上失败的依赖项链的描述。 | 引用对象，例如，"cascadeFailedOn":{ |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| | | "ref":"myRunnableObjectId"} |
| emrStepLog | 仅在尝试 EMR 活动时可用的 EMR 步骤日志 | String |
| errorId | 该对象失败时显示的 errorId。 | String |
| errorMessage | 该对象失败时显示的 errorMessage。 | String |
| errorStackTrace | 该对象失败时显示的错误堆栈跟踪。 | String |
| hadoopJobLog | 在尝试基于 EMR 的活动时可用的 Hadoop 任务日志。 | String |
| hostname | 已执行任务尝试的客户端的主机名。 | String |
| node | 将为其执行此先决条件的节点。 | 引用对象，例如，"node":{"ref":"myRunnableObjectId"} |
| reportProgressTime | 远程活动报告进度的最近时间。 | DateTime |
| @scheduledEndTime | 对象的计划结束时间。 | DateTime |
| @scheduledStartTime | 对象的计划开始时间。 | DateTime |
| @status | 该对象的状态。 | String |
| @version | 用来创建对象的管道版本。 | String |
| @waitingOn | 该对象在其上处于等待状态的依赖项列表的描述。 | 引用对象，例如，"waitingOn":{"ref":"myRunnableObjectId"} |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息 | String |
| @pipelineId | 该对象所属的管道的 ID | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象 | String |

### 另请参阅

- ShellCommandActivity
- 存在

## 数据库

以下是 AWS Data Pipeline 数据库对象：

Objects

- JdbcDatabase
- RdsDatabase
- RedshiftDatabase

## JdbcDatabase

定义 JDBC 数据库。

### 示例

以下是该对象类型的示例。

```
{
  "id" : "MyJdbcDatabase",
  "type" : "JdbcDatabase",
  "connectionString" : "jdbc:redshift://hostname:portnumber/dbname",
  "jdbcDriverClass" : "com.amazon.redshift.jdbc41.Driver",
```

```
  "jdbcDriverJarUri" : "s3://redshift-downloads/drivers/RedshiftJDBC41-1.1.6.1006.jar",
  "username" : "user_name",
  "*password" : "my_password"
}
```

## 语法

| 必填字段 | 描述 | 槽类型 |
| --- | --- | --- |
| connectionString | 用于访问数据库的 JDBC 连接字符串。 | String |
| jdbcDriverClass | 建立 JDBC 连接之前要加载的驱动程序类。 | String |
| *password | 提供的密码。 | String |
| username | 连接到数据库时提供的用户名。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| databaseName | 要附加到的逻辑数据库的名称 | String |
| jdbcDriverJarUri | 连接到数据库所用 JDBC 驱动程序 JAR 文件在 Amazon S3 中的位置。AWS Data Pipeline 必须具有读取此 JAR 文件的权限。 | String |
| jdbcProperties | 格式为 A=B 的一对值，设置作为此数据库的 JDBC 连接上的属性。 | String |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @version | 用于创建对象的管道版本。 | String |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息。 | String |
| @pipelineId | 该对象所属的管道的 ID。 | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象。 | String |

## RdsDatabase

定义 Amazon RDS 数据库。

> **Note**
>
> RdsDatabase 不支持 Aurora。把 the section called “JdbcDatabase” 用于 Aurora。

### 示例

以下是该对象类型的示例。

```
{
  "id" : "MyRdsDatabase",
  "type" : "RdsDatabase",
  "region" : "us-east-1",
  "username" : "user_name",
  "*password" : "my_password",
  "rdsInstanceId" : "my_db_instance_identifier"
}
```

对于 Oracle 引擎，jdbcDriverJarUri 字段是必填的，并且您可以指定以下驱动程序：http://www.oracle.com/technetwork/database/features/jdbc/jdbc-drivers-12c-download-1958347.html。对于 SQL Server 引擎，jdbcDriverJarUri 字段是必填的，并且您可以指定以下驱动程序：https://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=11774。对于 MySQL 和 PostgreSQL 引擎，jdbcDriverJarUri 字段是可选的。

## 语法

| 必填字段 | 描述 | 槽类型 |
| --- | --- | --- |
| *password | 提供的密码。 | String |
| rdsInstanceId | DB 实例的 `DBInstanceIdentifier` 属性。 | String |
| username | 连接到数据库时提供的用户名。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| databaseName | 要附加到的逻辑数据库的名称 | String |
| jdbcDriverJarUri | 连接到数据库所用 JDBC 驱动程序 JAR 文件在 Amazon S3 中的位置。AWS Data Pipeline 必须具有读取此 JAR 文件的权限。对于 MySQL 和 PostgreSQL 引擎，如果未指定此字段则使用默认的驱动程序，但您可以使用此字段覆盖默认值。对于 Oracle 和 SQL Server 引擎，此字段为必填字段。 | String |
| jdbcProperties | 格式为 A=B 的一对值，设置作为此数据库的 JDBC 连接上的属性。 | String |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |
| region | 数据库所在区域的代码。例如，us-east-1。 | String |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @version | 用于创建对象的管道版本。 | String |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息。 | String |
| @pipelineId | 该对象所属的管道的 ID。 | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象。 | String |

## RedshiftDatabase

定义 Amazon Redshift 数据库。`RedshiftDatabase` 表示由您的管道使用的数据库的属性。

### 示例

以下是该对象类型的示例。

```
{
  "id" : "MyRedshiftDatabase",
  "type" : "RedshiftDatabase",
  "clusterId" : "myRedshiftClusterId",
  "username" : "user_name",
  "*password" : "my_password",
  "databaseName" : "database_name"
}
```

默认情况下，此对象将使用 Postgres 驱动程序，这将需要 `clusterId` 字段。要使用 Amazon Redshift 驱动程序，请改为在 `connectionString` 字段中指定来自 Amazon Redshift 控制台的 Amazon Redshift 数据库连接字符串 (以“jdbc:redshift:”开头)。

### 语法

| 必填字段 | 描述 | 槽类型 |
| --- | --- | --- |
| *password | 提供的密码。 | String |
| username | 连接到数据库时提供的用户名。 | String |

| 所需的组 (下列选项之一是必需的) | 描述 | 槽类型 |
| --- | --- | --- |
| clusterId | 创建 Amazon Redshift 群集时用户提供的标识符。例如，如果您的 Amazon Redshift 群集的终端节点是 mydb.example.us-east-1.redshift.amazonaws.com，正确的标识符是 mydb。在 Amazon Redshift 控制台中，您可以从“Cluster Identifier”或“Cluster Name”获取此值。 | String |
| connectionString | 用于连接到由账户拥有的不同于管道的 Amazon Redshift 实例的 JDBC 终端节点。您不能同时指定 connectionString 和 clusterId 。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| databaseName | 要附加到的逻辑数据库的名称。 | String |
| jdbcProperties | 格式为 A=B 的一对值，设置为该数据库的 JDBC 连接上的属性。 | String |
| parent | 作为槽继承源的当前对象的父项。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |
| region | 数据库所在区域的代码。例如，us-east-1。 | 枚举 |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @version | 用于创建对象的管道版本。 | String |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息。 | String |
| @pipelineId | 该对象所属的管道的 ID。 | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象。 | String |

# 数据格式

以下是 AWS Data Pipeline 数据格式对象：

Objects

- CSV 数据格式
- 自定义数据格式
- DynamoDBDataFormat
- DynamoDBExportDataFormat
- RegEx 数据格式
- TSV 数据格式

## CSV 数据格式

一种逗号分隔的数据格式，其中列分隔符是逗号，记录分隔符是换行符。

### 示例

以下是该对象类型的示例。

```
{
  "id" : "MyOutputDataType",
  "type" : "CSV",
  "column" : [
    "Name STRING",
    "Score INT",
    "DateOfBirth TIMESTAMP"
  ]
```

```
}
```

### 语法

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| column | 具有该数据节点描述的数据的各个字段指定的数据类型的列名。例如：hostname STRING。对于多个值，使用空格分隔的列名和数据类型。 | String |
| escapeChar | 一个字符，例如“\”，指示分析器忽略下一个字符。 | String |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @version | 用来创建对象的管道版本。 | String |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息 | String |
| @pipelineId | 该对象所属的管道的 ID | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象 | String |

## 自定义数据格式

由特定的列分隔符、记录分隔符和转义字符组合定义的自定义数据格式。

## 示例

以下是该对象类型的示例。

```
{
  "id" : "MyOutputDataType",
  "type" : "Custom",
  "columnSeparator" : ",",
  "recordSeparator" : "\n",
  "column" : [
    "Name STRING",
    "Score INT",
    "DateOfBirth TIMESTAMP"
  ]
}
```

## 语法

| 必填字段 | 描述 | 槽类型 |
| --- | --- | --- |
| columnSeparator | 一个指示数据文件中的列结尾的字符。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| column | 具有该数据节点描述的数据的各个字段指定的数据类型的列名。例如：hostname STRING。对于多个值，使用空格分隔的列名和数据类型。 | String |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |
| recordSeparator | 一个指示数据文件中的行结尾的字符，例如“\n”。仅支持单个字符。 | String |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @version | 用来创建对象的管道版本。 | String |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息 | String |
| @pipelineId | 该对象所属的管道的 ID | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象 | String |

## DynamoDBDataFormat

将架构应用于 DynamoDB 表以使其可供 Hive 查询访问。DynamoDBDataFormat 与 HiveActivity 对象以及 DynamoDBDataNode 输入和输出一起使用。DynamoDBDataFormat 要求您在 Hive 查询中指定所有列。要更灵活地在 Hive 查询或 Amazon S3 支持中指定特定列，请参阅 DynamoDBExportDataFormat。

> **Note**
>
> DynamoDB 布尔类型不会映射到 Hive 布尔类型。不过，可以将 DynamoDB 整数值 0 或 1 映射到 Hive 布尔类型。

### 示例

以下示例说明如何使用 DynamoDBDataFormat 将架构分配给 DynamoDBDataNode 输入，这将允许 HiveActivity 对象按命名的列访问数据并将数据复制到 DynamoDBDataNode 输出。

```
{
  "objects": [
    {
      "id" : "Exists.1",
      "name" : "Exists.1",
      "type" : "Exists"
```

```
    },
    {
      "id" : "DataFormat.1",
      "name" : "DataFormat.1",
      "type" : "DynamoDBDataFormat",
      "column" : [
         "hash STRING",
        "range STRING"
      ]
    },
    {
      "id" : "DynamoDBDataNode.1",
      "name" : "DynamoDBDataNode.1",
      "type" : "DynamoDBDataNode",
      "tableName" : "$INPUT_TABLE_NAME",
      "schedule" : { "ref" : "ResourcePeriod" },
      "dataFormat" : { "ref" : "DataFormat.1" }
    },
    {
      "id" : "DynamoDBDataNode.2",
      "name" : "DynamoDBDataNode.2",
      "type" : "DynamoDBDataNode",
      "tableName" : "$OUTPUT_TABLE_NAME",
      "schedule" : { "ref" : "ResourcePeriod" },
      "dataFormat" : { "ref" : "DataFormat.1" }
    },
    {
      "id" : "EmrCluster.1",
      "name" : "EmrCluster.1",
      "type" : "EmrCluster",
      "schedule" : { "ref" : "ResourcePeriod" },
      "masterInstanceType" : "m1.small",
      "keyPair" : "$KEYPAIR"
    },
    {
      "id" : "HiveActivity.1",
      "name" : "HiveActivity.1",
      "type" : "HiveActivity",
      "input" : { "ref" : "DynamoDBDataNode.1" },
      "output" : { "ref" : "DynamoDBDataNode.2" },
      "schedule" : { "ref" : "ResourcePeriod" },
      "runsOn" : { "ref" : "EmrCluster.1" },
      "hiveScript" : "insert overwrite table ${output1} select * from ${input1} ;"
    },
```

```
    {
      "id" : "ResourcePeriod",
      "name" : "ResourcePeriod",
      "type" : "Schedule",
      "period" : "1 day",
      "startDateTime" : "2012-05-04T00:00:00",
      "endDateTime" : "2012-05-05T00:00:00"
    }
  ]
}
```

## 语法

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| column | 具有该数据节点描述的数据的各个字段指定的数据类型的列名。例如，hostname STRING。对于多个值，请使用以空格分隔的列名和数据类型。 | String |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @version | 用于创建对象的管道版本。 | String |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息。 | String |
| @pipelineId | 该对象所属的管道的 ID。 | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象。 | String |

# DynamoDBExportDataFormat

将架构应用于 DynamoDB 表以使其可供 Hive 查询访问。将 DynamoDBExportDataFormat 与 HiveCopyActivity 对象和 DynamoDBDataNode 或 S3DataNode 输入和输出一起使用。DynamoDBExportDataFormat 具有以下优势：

- 提供 DynamoDB 和 Amazon S3 支持
- 允许您在 Hive 查询中按特定列筛选数据
- 导出 DynamoDB 中的所有属性（即使您有一个稀疏架构）

> **Note**
>
> DynamoDB 布尔类型不会映射到 Hive 布尔类型。不过，可以将 DynamoDB 整数值 0 或 1 映射到 Hive 布尔类型。

## 示例

以下示例说明如何使用 HiveCopyActivity 和 DynamoDBExportDataFormat 将数据从一个 DynamoDBDataNode 复制到另一个 DynamoDBDataNode，并基于时间戳进行筛选。

```
{
  "objects": [
    {
      "id" : "DataFormat.1",
      "name" : "DataFormat.1",
      "type" : "DynamoDBExportDataFormat",
      "column" : "timeStamp BIGINT"
    },
    {
      "id" : "DataFormat.2",
      "name" : "DataFormat.2",
      "type" : "DynamoDBExportDataFormat"
    },
    {
      "id" : "DynamoDBDataNode.1",
      "name" : "DynamoDBDataNode.1",
      "type" : "DynamoDBDataNode",
      "tableName" : "item_mapped_table_restore_temp",
      "schedule" : { "ref" : "ResourcePeriod" },
```

```
      "dataFormat" : { "ref" : "DataFormat.1" }
    },
    {
      "id" : "DynamoDBDataNode.2",
      "name" : "DynamoDBDataNode.2",
      "type" : "DynamoDBDataNode",
      "tableName" : "restore_table",
      "region" : "us_west_1",
      "schedule" : { "ref" : "ResourcePeriod" },
      "dataFormat" : { "ref" : "DataFormat.2" }
    },
    {
      "id" : "EmrCluster.1",
      "name" : "EmrCluster.1",
      "type" : "EmrCluster",
      "schedule" : { "ref" : "ResourcePeriod" },
      "masterInstanceType" : "m1.xlarge",
      "coreInstanceCount" : "4"
    },
    {
      "id" : "HiveTransform.1",
      "name" : "Hive Copy Transform.1",
      "type" : "HiveCopyActivity",
      "input" : { "ref" : "DynamoDBDataNode.1" },
      "output" : { "ref" : "DynamoDBDataNode.2" },
      "schedule" : { "ref" : "ResourcePeriod" },
      "runsOn" : { "ref" : "EmrCluster.1" },
      "filterSql" : "`timeStamp` > unix_timestamp(\"#{@scheduledStartTime}\", \"yyyy-
MM-dd'T'HH:mm:ss\")"
    },
    {
      "id" : "ResourcePeriod",
      "name" : "ResourcePeriod",
      "type" : "Schedule",
      "period" : "1 Hour",
      "startDateTime" : "2013-06-04T00:00:00",
      "endDateTime" : "2013-06-04T01:00:00"
    }
  ]
}
```

### 语法

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| column | 具有该数据节点描述的数据的各个字段指定的数据类型的列名。例如：hostname STRING | String |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @version | 用来创建对象的管道版本。 | String |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息 | String |
| @pipelineId | 该对象所属的管道的 ID | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象 | String |

## RegEx 数据格式

由正则表达式定义的自定义数据格式。

### 示例

以下是该对象类型的示例。

```
{
  "id" : "MyInputDataType",
  "type" : "RegEx",
```

```
  "inputRegEx" : "([^ ]*) ([^ ]*) ([^ ]*) (-|\\[[^\\]]*\\]) ([^ \"]*|\"[^\"]*\") (-|
[0-9]*) (-|[0-9]*)(?: ([^ \"]*|\"[^\"]*\") ([^ \"]*|\"[^\"]*\"))?",
  "outputFormat" : "%1$s %2$s %3$s %4$s %5$s %6$s %7$s %8$s %9$s",
  "column" : [
    "host STRING",
    "identity STRING",
    "user STRING",
    "time STRING",
    "request STRING",
    "status STRING",
    "size STRING",
    "referer STRING",
    "agent STRING"
  ]
}
```

## 语法

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| column | 具有该数据节点描述的数据的各个字段指定的数据类型的列名。例如：hostname STRING。对于多个值，使用空格分隔的列名和数据类型。 | String |
| inputRegEx | 用于解析 S3 输入文件的正则表达式。inputRegEx 提供了一种方式来从文件中的相对非结构化数据检索列。 | String |
| outputFormat | 由 inputRegEx 检索但通过 Java 格式化程序语法引用为 %1$s %2$s 的列字段。 | String |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @version | 用来创建对象的管道版本。 | String |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息 | String |
| @pipelineId | 该对象所属的管道的 ID | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象 | String |

# TSV 数据格式

一种逗号分隔的数据格式，其中列分隔符是制表符，记录分隔符是换行符。

## 示例

以下是该对象类型的示例。

```
{
  "id" : "MyOutputDataType",
  "type" : "TSV",
  "column" : [
    "Name STRING",
    "Score INT",
    "DateOfBirth TIMESTAMP"
  ]
}
```

## 语法

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| column | 该数据节点描述的数据的列名和数据类型。例如，"Name STRING" 表示名为 Name 的列，其字段的数据类型为 STRING。请使用逗号分隔多个列名和数据类型对 (如示例中所示)。 | String |
| columnSeparator | 将一列中的字段与下一列中字段分隔的字符。默认为“\t”。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| escapeChar | 一个字符，例如“\”，指示分析器忽略下一个字符。 | String |
| parent | 作为槽继承源的当前对象的父项。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |
| recordSeparator | 分隔记录的字符。默认为“\n”。 | String |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @version | 用于创建对象的管道版本。 | String |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息。 | String |
| @pipelineId | 该对象所属的管道的 ID。 | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象。 | String |

# 操作

以下是 AWS Data Pipeline 操作对象：

Objects

- SnsAlarm
- 终止

# SnsAlarm

在活动失败或成功完成时发送 Amazon SNS 通知消息。

## 示例

以下是该对象类型的示例。node.input 和 node.output 的值来自在其 onSuccess 字段中引用该对象的数据节点或活动。

```
{
  "id" : "SuccessNotify",
  "name" : "SuccessNotify",
  "type" : "SnsAlarm",
  "topicArn" : "arn:aws:sns:us-east-1:28619EXAMPLE:ExampleTopic",
  "subject" : "COPY SUCCESS: #{node.@scheduledStartTime}",
  "message" : "Files were copied from #{node.input} to #{node.output}."
}
```

## 语法

| 必填字段 | 描述 | 槽类型 |
| --- | --- | --- |
| 消息 | Amazon SNS 通知的正文文本。 | String |
| role | 创建 Amazon SNS 警报所用的 IAM 角色。 | String |
| subject | Amazon SNS 通知消息的主题行。 | String |
| topicArn | 消息的目标 Amazon SNS 主题 ARN。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| node | 将为其执行此操作的节点。 | 引用对象，例如，"node":{"ref":"myRunnableObjectId"} |
| @version | 用来创建对象的管道版本。 | String |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息。 | String |
| @pipelineId | 该对象所属的管道的 ID。 | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象。 | String |

## 终止

一个用于触发取消挂起或未完成活动、资源或数据节点的操作。AWS Data Pipeline 会尝试将活动、资源或数据节点置于 CANCELLED 状态（如果它未由 `lateAfterTimeout` 值启动）。

您不能终止包含 `onSuccess`、`OnFail` 或 `onLateAction` 资源的操作。

### 示例

以下是该对象类型的示例。在此示例中，`MyActivity` 的 `onLateAction` 字段包含对操作 `DefaultAction1` 的引用。当您提供针对 `onLateAction` 的操作时，您还必须提供 `lateAfterTimeout` 值以指示自管道的计划开始后经过的时间段，在该时间段后，活动将被视为延迟。

```
{
  "name" : "MyActivity",
  "id" : "DefaultActivity1",
  "schedule" : {
    "ref" : "MySchedule"
```

```
  },
  "runsOn" : {
    "ref" : "MyEmrCluster"
  },
  "lateAfterTimeout" : "1 Hours",
  "type" : "EmrActivity",
  "onLateAction" : {
    "ref" : "DefaultAction1"
  },
  "step" : [
    "s3://myBucket/myPath/myStep.jar,firstArg,secondArg",
    "s3://myBucket/myPath/myOtherStep.jar,anotherArg"
  ]
},
{
  "name" : "TerminateTasks",
  "id" : "DefaultAction1",
  "type" : "Terminate"
}
```

## 语法

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| parent | 作为槽继承源的当前对象的父项。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| node | 将为其执行此操作的节点。 | 引用对象，例如，"node":{"ref":"myRunnableObjectId"} |
| @version | 用于创建对象的管道版本。 | String |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息。 | String |
| @pipelineId | 该对象所属的管道的 ID。 | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象。 | String |

# 计划

定义已计划事件的发生时间，例如何时活动运行。

> **Note**
>
> 如果计划的开始时间为过去时间，AWS Data Pipeline 将回填您的管道并在指定的开始时间立即开始计划运行。出于测试/开发目的，请使用一个相对较短的时间间隔。否则 AWS Data Pipeline 会尝试针对该时间间隔对所有管道运行进行排队和计划。如果管道组件 `scheduledStartTime` 早于 1 天前，AWS Data Pipeline 将通过阻止管道激活来尝试防止意外回填。

## 示例

以下是该对象类型的示例。它定义了一个每小时计划，开始时间为 2012-09-01 的 00:00:00，结束时间为 2012-10-01 的 00:00:00。第一个时段的结束时间为 2012-09-01 的 01:00:00。

```
{
  "id" : "Hourly",
  "type" : "Schedule",
  "period" : "1 hours",
  "startDateTime" : "2012-09-01T00:00:00",
  "endDateTime" : "2012-10-01T00:00:00"
}
```

以下管道将在 `FIRST_ACTIVATION_DATE_TIME` 启动，并每小时运行一次，直至 2014-04-25 的 22:00:00。

```
{
     "id": "SchedulePeriod",
     "name": "SchedulePeriod",
     "startAt": "FIRST_ACTIVATION_DATE_TIME",
     "period": "1 hours",
     "type": "Schedule",
     "endDateTime": "2014-04-25T22:00:00"
   }
```

以下管道将在 FIRST_ACTIVATION_DATE_TIME 启动，每小时运行一次，并在运行三次后完成。

```
{
     "id": "SchedulePeriod",
     "name": "SchedulePeriod",
     "startAt": "FIRST_ACTIVATION_DATE_TIME",
     "period": "1 hours",
     "type": "Schedule",
     "occurrences": "3"
   }
```

以下管道将在 2014-04-25 的 22:00:00 启动，每小时运行一次，并在运行三次后结束。

```
{
     "id": "SchedulePeriod",
     "name": "SchedulePeriod",
     "startDateTime": "2014-04-25T22:00:00",
     "period": "1 hours",
     "type": "Schedule",
     "occurrences": "3"
   }
```

按需 (使用默认对象)

```
{
  "name": "Default",
  "resourceRole": "DataPipelineDefaultResourceRole",
  "role": "DataPipelineDefaultRole",
  "scheduleType": "ondemand"
}
```

按需 (使用显式计划对象)

```
{
  "name": "Default",
  "resourceRole": "DataPipelineDefaultResourceRole",
  "role": "DataPipelineDefaultRole",
  "scheduleType": "ondemand"
},
{
  "name": "DefaultSchedule",
  "type": "Schedule",
  "id": "DefaultSchedule",
  "period": "ONDEMAND_PERIOD",
  "startAt": "ONDEMAND_ACTIVATION_TIME"
},
```

以下示例说明如何从默认对象继承计划、为该对象明确设置计划或通过父引用提供计划：

继承自默认对象的计划

```
{
  "objects": [
  {
      "id": "Default",
      "failureAndRerunMode":"cascade",
      "resourceRole": "DataPipelineDefaultResourceRole",
      "role": "DataPipelineDefaultRole",
      "pipelineLogUri": "s3://myLogsbucket",
      "scheduleType": "cron",
      "schedule": {
        "ref": "DefaultSchedule"
      }
   },
   {
      "type": "Schedule",
      "id": "DefaultSchedule",
      "occurrences": "1",
      "period": "1 Day",
      "startAt": "FIRST_ACTIVATION_DATE_TIME"
    },
    {
      "id": "A_Fresh_NewEC2Instance",
      "type": "Ec2Resource",
      "terminateAfter": "1 Hour"
    },
```

```
    {
      "id": "ShellCommandActivity_HelloWorld",
      "runsOn": {
        "ref": "A_Fresh_NewEC2Instance"
      },
      "type": "ShellCommandActivity",
      "command": "echo 'Hello World!'"
    }
  ]
}
```

对象上的显式计划

```
{
  "objects": [
  {
      "id": "Default",
      "failureAndRerunMode":"cascade",
      "resourceRole": "DataPipelineDefaultResourceRole",
      "role": "DataPipelineDefaultRole",
      "pipelineLogUri": "s3://myLogsbucket",
      "scheduleType": "cron"

   },
   {
      "type": "Schedule",
      "id": "DefaultSchedule",
      "occurrences": "1",
      "period": "1 Day",
      "startAt": "FIRST_ACTIVATION_DATE_TIME"
    },
    {
      "id": "A_Fresh_NewEC2Instance",
      "type": "Ec2Resource",
      "terminateAfter": "1 Hour"
    },
    {
      "id": "ShellCommandActivity_HelloWorld",
      "runsOn": {
        "ref": "A_Fresh_NewEC2Instance"
      },
      "schedule": {
        "ref": "DefaultSchedule"
```

```
      },
      "type": "ShellCommandActivity",
      "command": "echo 'Hello World!'"
    }
  ]
}
```

来自父引用的计划

```
{
  "objects": [
  {
      "id": "Default",
      "failureAndRerunMode":"cascade",
      "resourceRole": "DataPipelineDefaultResourceRole",
      "role": "DataPipelineDefaultRole",
      "pipelineLogUri": "s3://myLogsbucket",
      "scheduleType": "cron"

   },
   {
      "id": "parent1",
      "schedule": {
        "ref": "DefaultSchedule"
      }

   },
   {
      "type": "Schedule",
      "id": "DefaultSchedule",
      "occurrences": "1",
      "period": "1 Day",
      "startAt": "FIRST_ACTIVATION_DATE_TIME"
    },
    {
      "id": "A_Fresh_NewEC2Instance",
      "type": "Ec2Resource",
      "terminateAfter": "1 Hour"
    },
    {
      "id": "ShellCommandActivity_HelloWorld",
      "runsOn": {
        "ref": "A_Fresh_NewEC2Instance"
```

```
      },
      "parent": {
        "ref": "parent1"
      },
      "type": "ShellCommandActivity",
      "command": "echo 'Hello World!'"
    }
  ]
}
```

## 语法

| 必填字段 | 描述 | 槽类型 |
| --- | --- | --- |
| 周期 | 管道应运行的频率。格式为“N [minutes\|hours\|days\|weeks\|months]”，其中 N 是一个数字，后跟其中一个时间说明符。例如，“15 minutes”表示每 15 分钟运行一次管道。最短时段为 15 分钟，最长时段为 3 年。 | 周期 |

| 所需的组 (下列选项之一是必需的) | 描述 | 槽类型 |
| --- | --- | --- |
| startAt | 计划的管道运行的开始日期和时间。有效值为 FIRST_ACTIVATION_DATE_TIME，已弃用此值以支持创建按需管道。 | 枚举 |
| startDateTime | 计划运行的开始日期和时间。您必须使用 startDateTime 或 startAt，但不能同时使用。 | DateTime |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| endDateTime | 计划运行的结束日期和时间。必须是晚于 startDateTime 或 startAt 的值的日期和时间。默认行为是计划运行，直至管道关闭。 | DateTime |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| occurrences | 在管道激活后执行管道的次数。不能将 occurrences 与 endDateTime 结合使用。 | 整数 |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @version | 用来创建对象的管道版本。 | String |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息 | String |
| @firstActivationTime | 对象创建时间。 | DateTime |
| @pipelineId | 该对象所属的管道的 ID | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象 | String |

# 实用程序

以下实用程序对象配置其他管道对象：

主题

- ShellScriptConfig
- EmrConfiguration
- 属性

# ShellScriptConfig

与活动一起使用来为 preActivityTaskConfig 和 postActivityTaskConfig 运行 shell 脚本。该对象适用于 HadoopActivity、HiveActivity、HiveCopyActivity 和 PigActivity。您为该脚本指定一个 S3 URI 和一个参数列表。

## 示例

带参数的 ShellScriptConfig：

```
{
   "id" : "ShellScriptConfig_1”,
   "name" : “prescript”,
   "type" : "ShellScriptConfig",
   "scriptUri": “s3://my-bucket/shell-cleanup.sh”,
   "scriptArgument" : ["arg1","arg2"]
 }
```

## 语法

该对象包含以下字段。

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| parent | 作为槽继承源的当前对象的父项。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |
| scriptArgument | 用于 Shell 脚本的参数列表。 | String |
| scriptUri | Amazon S3 中需要下载并运行的脚本 URI。 | String |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @version | 用于创建对象的管道版本。 | String |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息。 | String |
| @pipelineId | 该对象所属的管道的 ID。 | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象。 | String |

## EmrConfiguration

EmrConfiguration 对象是用于 4.0.0 版或更高版本的 EMR 集群的配置。配置 (列表形式) 是一个用于 RunJobFlow API 调用的参数。Amazon EMR 的配置 API 采用分类和属性。AWS Data Pipeline 将 EmrConfiguration 与相应的属性对象结合使用来配置 EmrCluster 应用程序，例如在管道执行中启动的 EMR 集群上的 Hadoop、Hive、Spark 或 Pig。由于只能为新集群更改配置，因此，您无法为现有资源提供 EmrConfiguration 对象。有关更多信息，请参阅https://docs.aws.amazon.com/ElasticMapReduce/latest/ReleaseGuide/。

### 示例

以下配置对象在 `core-site.xml` 中设置 `io.file.buffer.size` 和 `fs.s3.block.size` 属性：

```
[
   {
      "classification":"core-site",
      "properties":
      {
         "io.file.buffer.size": "4096",
         "fs.s3.block.size": "67108864"
      }
   }
]
```

相应的管道对象定义在 `property` 字段中使用一个 EmrConfiguration 对象和一系列属性对象：

```
{
  "objects": [
    {
      "name": "ReleaseLabelCluster",
```

```
      "releaseLabel": "emr-4.1.0",
      "applications": ["spark", "hive", "pig"],
      "id": "ResourceId_I1mCc",
      "type": "EmrCluster",
      "configuration": {
        "ref": "coresite"
      }
    },
    {
      "name": "coresite",
      "id": "coresite",
      "type": "EmrConfiguration",
      "classification": "core-site",
      "property": [{
        "ref": "io-file-buffer-size"
      },
      {
        "ref": "fs-s3-block-size"
      }
      ]
    },
    {
      "name": "io-file-buffer-size",
      "id": "io-file-buffer-size",
      "type": "Property",
      "key": "io.file.buffer.size",
      "value": "4096"
    },
    {
      "name": "fs-s3-block-size",
      "id": "fs-s3-block-size",
      "type": "Property",
      "key": "fs.s3.block.size",
      "value": "67108864"
    }
  ]
}
```

以下示例是一个嵌套配置，用于通过 hadoop-env 分类设置 Hadoop 环境：

```
[
  {
    "classification": "hadoop-env",
```

```
    "properties": {},
    "configurations": [
      {
        "classification": "export",
        "properties": {
          "YARN_PROXYSERVER_HEAPSIZE": "2396"
        }
      }
    ]
  }
]
```

以下是使用此配置的相应管道定义对象：

```
{
  "objects": [
    {
      "name": "ReleaseLabelCluster",
      "releaseLabel": "emr-4.0.0",
      "applications": ["spark", "hive", "pig"],
      "id": "ResourceId_I1mCc",
      "type": "EmrCluster",
      "configuration": {
        "ref": "hadoop-env"
      }
    },
    {
      "name": "hadoop-env",
      "id": "hadoop-env",
      "type": "EmrConfiguration",
      "classification": "hadoop-env",
      "configuration": {
        "ref": "export"
      }
    },
    {
      "name": "export",
      "id": "export",
      "type": "EmrConfiguration",
      "classification": "export",
      "property": {
        "ref": "yarn-proxyserver-heapsize"
      }
```

```
    },
    {
      "name": "yarn-proxyserver-heapsize",
      "id": "yarn-proxyserver-heapsize",
      "type": "Property",
      "key": "YARN_PROXYSERVER_HEAPSIZE",
      "value": "2396"
    },
  ]
}
```

以下示例修改了 EMR 集群的 Hive 特定属性：

```
{
    "objects": [
        {
            "name": "hivesite",
            "id": "hivesite",
            "type": "EmrConfiguration",
            "classification": "hive-site",
            "property": [
                {
                    "ref": "hive-client-timeout"
                }
            ]
        },
        {
            "name": "hive-client-timeout",
            "id": "hive-client-timeout",
            "type": "Property",
            "key": "hive.metastore.client.socket.timeout",
            "value": "2400s"
        }
    ]
}
```

## 语法

该对象包含以下字段。

| 必填字段 | 描述 | 槽类型 |
| --- | --- | --- |
| 分类 | 配置的分类。 | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| 配置 | 此配置的子配置。 | 引用对象，例如，"configuration":{"ref":"myEmrConfigurationId"} |
| parent | 槽将继承自的当前对象的父级。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |
| property | 配置属性。 | 引用对象，例如，"property":{"ref":"myPropertyId"} |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @version | 用来创建对象的管道版本。 | String |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息 | String |
| @pipelineId | 该对象所属的管道的 ID | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象 | String |

### 另请参阅

- EmrCluster
- 属性
- Amazon EMR 版本指南

## 属性

用于 EmrConfiguration 对象的单个键/值属性。

### 示例

以下管道定义显示一个 EmrConfiguration 对象和用于启动 EmrCluster 的相应属性对象：

```
{
  "objects": [
    {
      "name": "ReleaseLabelCluster",
      "releaseLabel": "emr-4.1.0",
      "applications": ["spark", "hive", "pig"],
      "id": "ResourceId_I1mCc",
      "type": "EmrCluster",
      "configuration": {
        "ref": "coresite"
      }
    },
    {
      "name": "coresite",
      "id": "coresite",
      "type": "EmrConfiguration",
      "classification": "core-site",
      "property": [{
        "ref": "io-file-buffer-size"
      },
      {
        "ref": "fs-s3-block-size"
      }
      ]
    },
    {
      "name": "io-file-buffer-size",
```

```
      "id": "io-file-buffer-size",
      "type": "Property",
      "key": "io.file.buffer.size",
      "value": "4096"
    },
    {
      "name": "fs-s3-block-size",
      "id": "fs-s3-block-size",
      "type": "Property",
      "key": "fs.s3.block.size",
      "value": "67108864"
    }
  ]
}
```

## 语法

该对象包含以下字段。

| 必填字段 | 描述 | 槽类型 |
| --- | --- | --- |
| key | key | String |
| value | value | String |

| 可选字段 | 描述 | 槽类型 |
| --- | --- | --- |
| parent | 作为槽继承源的当前对象的父项。 | 引用对象，例如，"parent":{"ref":"myBaseObjectId"} |

| 运行时字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @version | 用于创建对象的管道版本。 | String |

| 系统字段 | 描述 | 槽类型 |
| --- | --- | --- |
| @error | 用于描述格式不正确的对象的错误消息。 | String |
| @pipelineId | 该对象所属的管道的 ID。 | String |
| @sphere | 对象的范围指明对象在生命周期中的位置：组件对象产生实例对象，后者执行尝试对象。 | String |

## 另请参阅

- EmrCluster
- EmrConfiguration
- Amazon EMR 版本指南

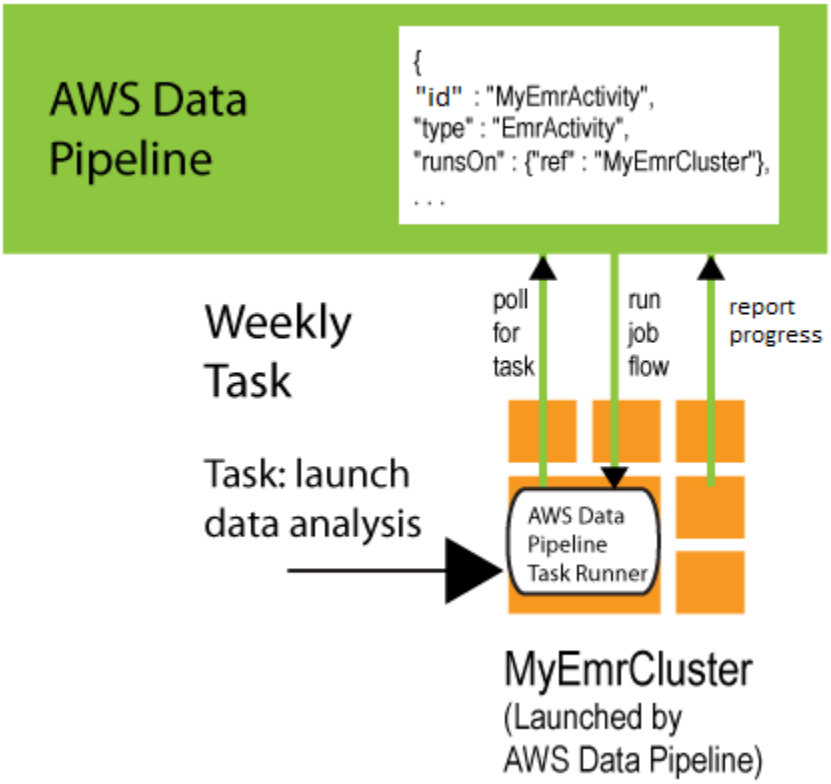
# 使用任务运行程序

任务运行程序是一种任务代理应用程序，此应用程序将轮询 AWS Data Pipeline 以获取计划任务，并在 Amazon EC2 实例、Amazon EMR 集群或其他计算资源上执行这些计划任务，还会在执行过程中报告状态。根据您的应用程序，您可以选择：

- 允许 AWS Data Pipeline 为您安装和管理一个或多个任务运行程序。在激活管道后，将自动创建由活动 runsOn 字段引用的默认 `Ec2Instance` 或 `EmrCluster` 对象。AWS Data Pipeline 负责在 EC2 实例上或 EMR 集群的主节点上安装任务运行程序。在此模式中，AWS Data Pipeline 可为您执行大多数实例或集群管理工作。
- 在您管理的资源上运行管道的全部或部分内容。可能的资源包括长时间运行的 Amazon EC2 实例、Amazon EMR 集群或物理服务器。您可以在几乎任何位置安装任务运行程序（可以是任务运行程序或您设备的自定义任务代理），前提是它可与 AWS Data Pipeline Web 服务进行通信。在此模式中，您几乎可以完全控制要使用的资源及其管理方式，并且您必须手动安装和配置任务运行程序。为此，请使用此部分中的过程，如使用任务运行程序在现有资源上执行工作中所述。

## AWS Data Pipeline 托管资源上的任务运行程序

当 AWS Data Pipeline 启动和管理某项资源时，Web 服务会自动在该资源上安装任务运行程序以处理管道中的任务。您为活动对象的 `runs0n` 字段指定计算资源（Amazon EC2 实例或 Amazon EMR 集群）。当 AWS Data Pipeline 启动此资源时，它将在该资源上安装任务运行程序并进行配置，以处理其 `runs0n` 字段设置为该资源的所有活动对象。当 AWS Data Pipeline 终止此资源时，任务运行程序日志将在资源关闭前发布到 Amazon S3 位置。




例如，如果您在管道中使用 EmrActivity 并在 runsOn 字段中指定 EmrCluster 资源，当 AWS Data Pipeline 处理此活动时，它将启动 Amazon EMR 集群并在主节点上安装任务运行程序。随后，此任务运行程序将处理其 runsOn 字段设置为该 EmrCluster 对象的活动的任务。来自管道定义的以下摘录说明了两个对象之间的此关系。

```
{
  "id" : "MyEmrActivity",
  "name" : "Work to perform on my data",
  "type" : "EmrActivity",
  "runsOn" : {"ref" : "MyEmrCluster"},
  "preStepCommand" : "scp remoteFiles localFiles",
  "step" : "s3://myBucket/myPath/myStep.jar,firstArg,secondArg",
  "step" : "s3://myBucket/myPath/myOtherStep.jar,anotherArg",
  "postStepCommand" : "scp localFiles remoteFiles",
  "input" : {"ref" : "MyS3Input"},
  "output" : {"ref" : "MyS3Output"}
},
{
  "id" : "MyEmrCluster",
  "name" : "EMR cluster to perform the work",
  "type" : "EmrCluster",
```

```
  "hadoopVersion" : "0.20",
  "keypair" : "myKeyPair",
  "masterInstanceType" : "m1.xlarge",
  "coreInstanceType" : "m1.small",
  "coreInstanceCount" : "10",
  "taskInstanceType" : "m1.small",
  "taskInstanceCount": "10",
  "bootstrapAction" : "s3://elasticmapreduce/libs/ba/configure-hadoop,arg1,arg2,arg3",
  "bootstrapAction" : "s3://elasticmapreduce/libs/ba/configure-other-stuff,arg1,arg2"
}
```

有关运行此活动的信息和示例，请参阅 EmrActivity。

如果您的管道中有多个 AWS Data Pipeline 托管的资源，则将在每个资源上安装任务运行程序，并且这些资源都将轮询 AWS Data Pipeline 以获取要处理的任务。

## 使用任务运行程序在现有资源上执行工作

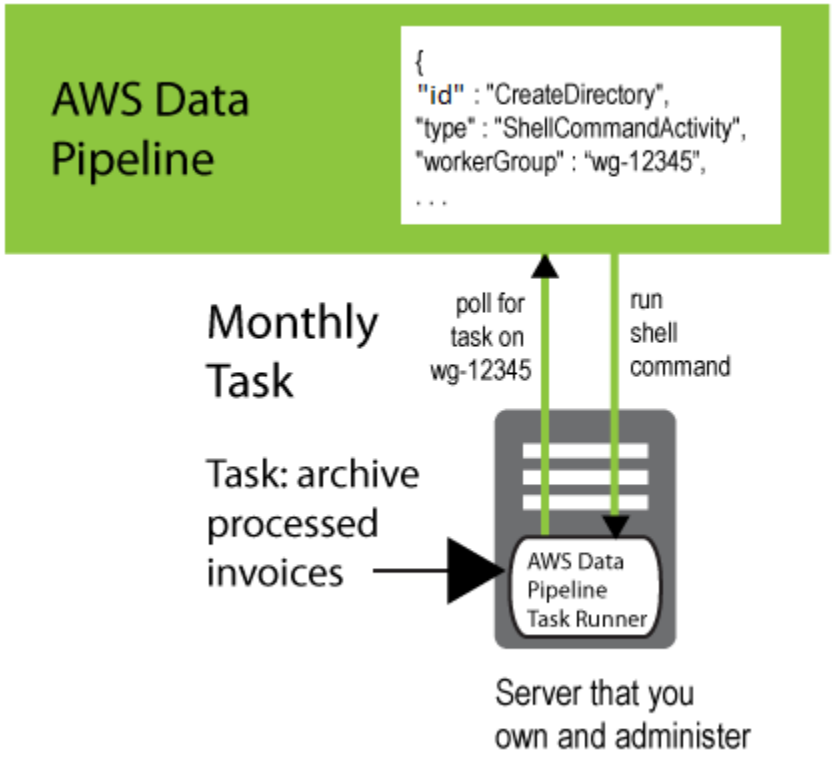
您可以在自己管理的计算资源（如 Amazon EC2 实例、物理服务器或工作站）上安装任务运行程序。可在任意位置的任何兼容的硬件或操作系统上安装任务运行程序，前提是它能够与 AWS Data Pipeline Web 服务进行通信。

例如，当您需要使用 AWS Data Pipeline 处理存储在组织防火墙内的数据时，此方法会很有用。通过在局域网内的服务器上安装任务运行程序，您可以安全访问本地数据库，然后轮询 AWS Data Pipeline 以获取要运行的下一个任务。当 AWS Data Pipeline 终止处理或删除管道时，任务运行程序实例仍将在计算资源上运行，直至您手动将其关闭。管道执行完成后，任务运行程序日志将保留。

要在您管理的资源上使用任务运行程序，您必须先下载任务运行程序，然后使用此部分中的过程将它安装到您的计算资源上。

> **Note**
>
> 您只能在 Linux、UNIX 或 macOS 上安装任务运行程序。Windows 操作系统不支持任务运行程序。
>
> 要使用任务运行程序 2.0，所需的最低 Java 版本为 1.7。

要将安装的任务运行程序连接到它将处理的管道活动，请将 `workerGroup` 字段添加到该对象，并将任务运行程序配置为轮询此工作线程组值。在运行任务运行程序 JAR 文件时，可通过将工作线程组字符串作为参数（例如 `--workerGroup=wg-12345`）传递来执行此操作。




```
{
  "id" : "CreateDirectory",
  "type" : "ShellCommandActivity",
  "workerGroup" : "wg-12345",
  "command" : "mkdir new-directory"
}
```

## 安装任务运行程序

此部分介绍如何安装和配置任务运行程序及其先决条件。安装是一个简单的手动过程。

安装任务运行程序

1. 任务运行程序需要 Java 版本 1.6 或 1.8。要确定是否已安装 Java 以及所运行的版本，请使用以下命令：

   ```
   java -version
   ```

如果您的计算机上未安装 Java 1.6 或 1.8，则可从 http://www.oracle.com/technetwork/java/index.html 下载其中一个版本。下载并安装 Java，然后继续下一步骤。

2. 从 https://s3.amazonaws.com/datapipeline-us-east-1/us-east-1/software/latest/TaskRunner/TaskRunner-1.0.jar 下载 `TaskRunner-1.0.jar`，然后将其复制到目标计算资源上的文件夹中。对于运行 `EmrActivity` 任务的 Amazon EMR 集群，请在集群的主节点上安装任务运行程序。
3. 使用任务运行程序连接到 AWS Data Pipeline Web 服务以处理您的命令时，用户需要以编程方式访问有权创建或管理数据管道的角色。有关更多信息，请参阅授权以编程方式访问。
4. 任务运行程序使用 HTTPS 连接到 AWS Data Pipeline Web 服务。如果您使用的是 AWS 资源，请确保在适当的路由表和子网 ACL 中启用 HTTPS。如果您使用的是防火墙或代理，请确保端口 443 处于打开状态。

## （可选）授予任务运行程序对 Amazon RDS 的访问权限

Amazon RDS 允许您使用数据库安全组控制对您的数据库实例的访问。数据库安全组与防火墙的功能类似，用于控制对您的数据库实例的网络访问。默认情况下，将为您的数据库实例关闭网络访问。您必须修改数据库安全组以让任务运行程序访问您的 Amazon RDS 实例。任务运行程序从运行时所在的实例获取 Amazon RDS 访问权，因此，您添加到 Amazon RDS 实例的账户和安全组取决于您将任务运行程序安装到的位置。

在 EC2-Classic 中授予对任务运行程序的访问权

1. 打开 Amazon RDS 控制台。
2. 在导航窗格中，选择 Instances，然后选择数据库实例。
3. 在安全与网络下面，选择安全组，将打开安全组页并选定该数据库安全组。选择数据库安全组的详细信息图标。
4. 在 Security Group Details 下，创建带适当的 Connection Type 和 Details 的规则。这些字段取决于任务运行程序运行的位置，如此处所述：

   - `Ec2Resource`
     - Connection Type：`EC2 Security Group`

       Details：*`my-security-group-name`* (您为 EC2 实例创建的安全组的名称)
   - `EmrResource`
     - Connection Type：`EC2 Security Group`

详细信息: ElasticMapReduce-master

- Connection Type：EC2 Security Group

详细信息: ElasticMapReduce-slave

- 您的本地环境 (本地)
  - Connection Type：CIDR/IP

    Details：*my-ip-address* (您的计算机的 IP 地址或您的网络的 IP 地址范围 (如果您的计算机位于防火墙之后))

5. 单击 Add (添加)。

在 EC2-Classic 中授予对任务运行程序的访问权

1. 打开 Amazon RDS 控制台。
2. 在导航窗格中，选择实例。
3. 选择数据库实例的详细信息图标。在安全与网络下，打开安全组的链接，这将使您转至 Amazon EC2 控制台。如果安全组使用旧控制台设计，请选择在控制台页面顶部显示的图标以切换到新控制台设计。
4. 在 Inbound 选项卡上，依次选择 Edit 和 Add Rule。指定在启动数据库实例时使用的数据库端口。来源取决于任务运行程序运行的位置，如此处所述：

   - Ec2Resource
     - *my-security-group-id* (您为 EC2 实例创建的安全组的 ID)
   - EmrResource
     - *master-security-group-id* (ElasticMapReduce-master 安全组的 ID)
     - *slave-security-group-id* (ElasticMapReduce-slave 安全组的 ID)
   - 您的本地环境 (本地)
     - *ip-address* (您的计算机的 IP 地址或您的网络的 IP 地址范围 (如果您的计算机位于防火墙之后))
5. 单击保存。

## 启动任务运行程序

在设置为已将任务运行程序安装到的目录的新命令提示窗口中，使用以下命令启动任务运行程序。

```
java -jar TaskRunner-1.0.jar --config ~/credentials.json --workerGroup=myWorkerGroup --
region=MyRegion --logUri=s3://mybucket/foldername
```

--config 选项指向您的凭证文件。

--workerGroup 选项指定工作线程组的名称，它必须是您在管道中为要处理的任务指定的同一值。

--region 选项指定从中提取要执行的任务的服务区域。

--logUri 选项用于将压缩的日志推送到 Amazon S3 中的某个位置。

当任务运行程序处于活动状态时，它将在终端窗口中输出日志文件的写入位置的路径。以下是示例。

```
Logging to /Computer_Name/.../output/logs
```

任务运行程序应独立于登录 shell 运行。如果您使用终端应用程序来连接到您的计算机，则可能需要使用实用工具 (如 nohup 或 screen) 来防止任务运行程序应用程序在您注销时退出。有关命令行选项的更多信息，请参阅任务运行程序配置选项。

### 验证任务运行程序日志记录

验证任务运行程序是否正常工作的最简单方法是，检查它是否在写入日志文件。任务运行程序每小时将日志文件写入到安装任务运行程序的目录下的 output/logs 目录中。文件名为 Task Runner.log.YYYY-MM-DD-HH，其中 HH 的范围为 00 到 23 (用 UDT 表示)。为了节省存储空间，将使用 GZip 压缩 8 小时之前的任何日志文件。

## 任务运行程序线程和先决条件

任务运行程序对每个任务、活动和先决条件使用一个线程池。--tasks 的默认设置为 2，这意味着，从任务池中分配 2 个线程，每个线程轮询 AWS Data Pipeline 服务以查找新任务。因此，--tasks 是一个性能优化属性，可用于帮助优化管道吞吐量。

先决条件的管道重试逻辑在任务运行程序中实施。将分配两个先决条件线程以轮询 AWS Data Pipeline 来获取先决条件对象。任务运行程序采用您在先决条件上定义的先决条件对象 retryDelay 和 preconditionTimeout 字段。

在许多情况下，减少先决条件轮询超时和重试次数有助于提高应用程序的性能。同样，具有长时间运行的先决条件的应用程序可能需要增大超时和重试值。有关先决条件对象的更多信息，请参阅先决条件。

# 任务运行程序配置选项

这些是在您启动任务运行程序时可从命令行使用的配置选项。

| 命令行参数 | 描述 |
| --- | --- |
| `--help` | 命令行帮助。示例：`Java -jar TaskRunner-1.0.jar --help` |
| `--config` | `credentials.json` 文件的路径和文件名。 |
| `--accessId` | 在发出请求时可由任务运行程序使用的 AWS 访问密钥 ID。<br><br>`--accessID` 和 `--secretKey` 选项提供了使用 credentials.json 文件的替代方法。如果还提供了 `credentials.json` 文件，则 `--accessID` 和 `--secretKey` 选项优先。 |
| `--secretKey` | 在发出请求时可由任务运行程序使用的 AWS 私有密钥。有关更多信息，请参阅`--accessID` 。 |
| `--endpoint` | 终端节点是作为 Web 服务入口点的 URL。从中发出请求的区域中的 AWS Data Pipeline 服务终端节点。可选。通常，指定区域已足够，您不需要设置终端节点。有关 AWS Data Pipeline 区域和终端节点的列表，请参阅 AWS 一般参考 中的 AWS Data Pipeline 区域和终端节点。 |
| `--workerGroup` | 任务运行程序检索其工作的工作线程组的名称。必需。<br><br>当任务运行程序轮询 Web 服务时，它使用您提供的凭证和 `workerGroup` 的值来选择要检索的任务（如果有）。您可以使用对您有意义的任何名称；唯一的要求是，字符串必须在任务运行程序和其相应的管道活动之间匹配。工作线程组名称将绑定到区域。即使在其他区域中具有完全 |

| 命令行参数 | 描述 |
| --- | --- |
| | 相同的工作线程组名称，任务运行程序也会始终从 --region 中指定的区域获取任务。 |
| --taskrunnerId | 报告进度时要使用的任务运行程序的 ID。可选。 |
| --output | 日志输出文件的任务运行程序目录。可选。日志文件存储在本地目录中，直至它们被推送到 Amazon S3。该选项覆盖默认目录。 |
| --region | 要使用的 区域。(可选) 但建议始终设置区域。如果您未指定区域，则任务运行程序将从默认服务区域（即 us-east-1 ）检索任务。<br><br>其他支持的区域为：eu-west-1 、ap-northeast-1 、ap-southeast-2 、us-west-2 。 |
| --logUri | 任务运行程序用来每小时备份日志文件的 Amazon S3 目标路径。在任务运行程序终止时，本地目录中的活动日志将推送到 Amazon S3 目标文件夹。 |
| --proxyHost | 任务运行程序客户端用于连接到 Amazon Web Services 的代理的主机。 |
| --proxyPort | 任务运行程序客户端用于连接到 Amazon Web Services 的代理主机的端口。 |
| --proxyUsername | 代理的用户名。 |
| --proxyPassword | 代理的密码。 |
| --proxyDomain | NTLM 代理的 Windows 域名。 |
| --proxyWorkstation | NTLM 代理的 Windows 工作站名。 |

# 将任务运行程序与代理结合使用

如果您使用的是代理主机，则可在调用任务运行程序指定其配置或设置环境变量 HTTPS_PROXY。用于任务运行程序的环境变量接受用于 AWS 命令行界面的相同配置。

# 任务运行程序和自定义 AMI

在您为管道指定 `Ec2Resource` 对象时，AWS Data Pipeline 将使用 AMI（用于为您安装和配置任务运行程序）为您创建 EC2 实例。在这种情况下，需要使用与 PV 兼容的实例类型。或者，您可以使用任务运行程序创建一个自定义 AMI，然后使用 `Ec2Resource` 对象的 `imageId` 字段指定此 AMI 的 ID。有关更多信息，请参阅Ec2Resource。

自定义 AMI 必须满足以下要求，AWS Data Pipeline 才能将它成功用于任务运行程序：

- 在运行实例的同一区域中创建 AMI。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的创建您自己的 AMI。
- 确保 AMI 的虚拟化类型受您计划使用的实例类型的支持。例如，I2 和 G2 实例类型需要 HVM AMI，而 T1、C1、M1 和 M2 实例类型需要 PV AMI。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的 Linux AMI 虚拟化类型。
- 安装以下软件：
  - Linux
  - Bash
  - wget
  - unzip
  - Java 1.6 或 1.8
  - cloud-init
- 创建和配置名为 `ec2-user` 的用户。

# 故障排除

如果 AWS Data Pipeline 出现问题，最常见的症状是管道无法运行。您可以使用控制台和 CLI 提供的数据来确定问题和查找解决方案。

目录



## 找出管道中的错误

AWS Data Pipeline 控制台是一个方便的工具，可以直观地监控管道的状态并轻松找出与管道运行失败或未完成相关的任何错误。

使用控制台找出有关运行失败或未完成的错误

1. 在列出管道页面上，如果任意管道实例的状态列显示已完成之外的状态，则管道在等待满足某个先决条件或者已经失败，您需要对管道进行问题排查。
2. 在列出管道页上，找到实例管道，然后选择左侧的三角形以展开详细信息。
3. 在该面板底部，选择查看执行详细信息；将打开实例摘要面板以显示选定实例的详细信息。
4. 在 Instance summary (实例摘要) 窗格中，选择实例旁的三角形可查看该实例的附加信息，然后选择 Details (详细信息)、More...(更多…) 如果所选实例的状态为 FAILED，则详细信息框中有错误消息条目、`errorStackTrace` 和其他信息。您可以将此信息保存到文件中。选择 OK（确定）。
5. 在实例摘要窗格中，选择尝试以查看每个尝试行的详细信息。
6. 要对未完成或失败的实例执行操作，请选中实例旁边的复选框。这将激活操作。然后，选择一个操作 (`Rerun|Cancel|Mark Finished`)。

# 确定服务于您的管道的 Amazon EMR 集群

如果 EMRCluster 或 EMRActivity 失败，并且 AWS Data Pipeline 控制台提供的错误信息不明确，您可以使用 Amazon EMR 控制台确定服务于您的管道的 Amazon EMR 集群。这可帮助您查找 Amazon EMR 提供的日志，以获取有关所发生错误的更多详细信息。

查看更详细的 Amazon EMR 错误信息

1. 在 AWS Data Pipeline 控制台中，选择管道实例旁边的三角形以展开实例详细信息。
2. 选择查看执行详细信息，然后选择组件旁边的三角形。
3. 在详细信息列中，选择更多...。信息屏幕随即打开，其中列出组件详细信息。从屏幕中找到并复制 instanceParent 值，例如：@EmrActivityId_xiFDD_2017-09-30T21:40:13
4. 导航到 Amazon EMR 控制台，搜索名称中包含匹配的 instanceParent 值的集群，然后选择调试。

   > **Note**
   >
   > 要使 Debug 按钮正常使用，您的管道定义必须将 EmrActivity enableDebugging 选项设置为 true，将 EmrLogUri 选项设置为有效路径。

5. 现在您知道哪些 Amazon EMR 集群包含导致您管道失败的错误，请按照 Amazon EMR 开发人员指南中的问题排查提示操作。

# 解释管道状态详细信息

AWS Data Pipeline 控制台和 CLI 中显示的不同状态级别指示管道及其组件的状况。管道状态仅仅是管道的概览；要查看更多信息，请查看单个管道组件的状态。您可以在控制台中不断单击管道来查看，或者使用 CLI 检索管道组件详细信息。

状态代码

ACTIVATING

组件或资源正在启动，例如 EC2 实例。

CANCELED

该组件在可以运行之前已被用户或 AWS Data Pipeline 取消。当该组件所依赖的其他组件或资源发生故障时，可能会自动发生这种情况。

CASCADE_FAILED

该组件或资源因其依赖项之一的级联故障而被取消，但该组件可能不是故障的原始来源。

DEACTIVATING

管道正在停用。

FAILED

组件或资源遇到错误并停止了工作。当某个组件或资源出现故障时，可能会导致取消和故障级联到依赖该组件的其他组件。

FINISHED

该组件完成了其分配到的工作。

INACTIVE

管道已停用。

PAUSED

该组件已暂停，目前未执行其工作。

PENDING

管道已准备好第一次激活。

RUNNING

资源正在运行并准备好接收工作。

SCHEDULED

资源已计划运行。

SHUTTING_DOWN

资源在成功完成其工作后正在关闭。

SKIPPED

在激活管道后，该组件使用晚于当前计划的时间戳跳过了执行间隔。

TIMEDOUT

该资源超过了`terminateAfter`阈值并已被AWS Data Pipeline停止。资源达到此状态后，AWS Data Pipeline忽略该资源的`actionOnResourceFailure`、`retryDelay`和`retryTimeout`值。该状态仅适用于资源。

`VALIDATING`

AWS Data Pipeline 正在验证管道定义。

`WAITING_FOR_RUNNER`

该组件正在等待其工作客户端检索工作项。组件和工作客户端关系由该组件定义的 `runsOn` 或 `workerGroup` 字段控制。

`WAITING_ON_DEPENDENCIES`

在执行其工作之前，该组件正在验证其默认和用户配置的前提条件是否得到满足。

# 查找错误日志

此部分说明如何查找 AWS Data Pipeline 写入的各种日志，您可以使用这些日志来确定特定故障和错误的源头。

## 管道日志

我们建议您配置管道在持久性位置中创建日志文件，如以下示例中，您在管道的 `Default` 对象上使用 `pipelineLogUri` 字段，导致所有管道组件默认使用 Amazon S3 日志位置（您可以在特定管道组件中配置日志位置来覆盖此项）。

> **Note**
>
> 任务运行程序默认将其日志存储在另一位置，在管道完成并且运行任务运行程序的实例终止时，该位置可能不可用。有关更多信息，请参阅验证任务运行程序日志记录。

要使用 AWS Data Pipeline CLI 在管道 JSON 文件中配置日志位置，请使用以下文本作为管道文件的开头：

```
{ "objects": [
{
  "id":"Default",
  "pipelineLogUri":"s3://mys3bucket/error_logs"
},
...
```

在您配置了管道日志目录之后，任务运行程序在目录中创建日志的副本，使用与之前章节中所述的有关任务运行程序日志的相同格式设置和文件名称。

## Hadoop 作业和 Amazon EMR 步骤日志

对于任何基于 Hadoop 的活动，例如 HadoopActivity、HiveActivity 或 PigActivity，您可以在运行时槽 hadoopJobLog 中返回的位置查看 Hadoop 任务日志。 EmrActivity 有自己的日志记录功能，这些日志使用 Amazon EMR 选择的位置存储，由运行时槽 emrStepLog 返回。有关更多信息，请参见《Amazon EMR 开发人员指南》的查看日志文件。

# 解决常见问题

本主题介绍了 AWS Data Pipeline 问题的各种症状，并推荐了步骤来解决这些问题。

目录



## 管道停滞在 Pending 状态

管道停滞在 PENDING 状态，指示该管道尚未激活，或者由于管道定义中的错误而激活失败。请确保您在使用 AWS Data Pipeline CLI 提交管道时，或在使用 AWS Data Pipeline 控制台尝试保存或激活管道时未收到任何错误。此外，请确保您的管道具有有效定义。

使用 CLI 在屏幕上查看管道定义：

```
aws datapipeline --get-pipeline-definition --pipeline-id df-EXAMPLE_PIPELINE_ID
```

确保管道定义完整，检查右大括号，验证所需逗号，检查缺少的引用以及其他语法错误。最好使用可直观地验证 JSON 文件语法的文本编辑器。

## 管道组件停滞在 Waiting for Runner 状态

如果您的管道处于 SCHEDULED 状态并且一个或多个任务显示为停滞在 WAITING_FOR_RUNNER 状态，请确保您为这些任务的 runsOn 或 workerGroup 字段设置了有效值。如果两个值均为空或缺失，则任务将无法启动，因为任务与执行任务的工作线程之间没有关联。在这种情况下，您已定义了工作，但未定义哪个计算机执行该工作。如果适用，请验证分配到管道组件的 workerGroup 值与您为任务运行程序配置的 workerGroup 值具有完全相同的名称和大小写。

> **Note**
>
> 如果您提供 runsOn 值并且存在 workerGroup，则将忽略 workerGroup。

此问题的另一个可能的原因是，向任务运行程序提供的终端节点和访问密钥，与 AWS Data Pipeline 控制台或安装 AWS Data Pipeline CLI 工具的计算机上的不同。您可能已经创建了没有可见错误的新管道，但任务运行程序由于凭证中的不同而轮询了错误位置，或者轮询了正确的位置但权限不足，无法确定和执行由管道定义指定的工作。

## 管道组件停滞在 WAITING_ON_DEPENDENCIES 状态

如果您的管道处于 SCHEDULED 状态并且一个或多个任务停滞在 WAITING_ON_DEPENDENCIES 状态，请确保已满足管道的初始先决条件。如果不满足逻辑链中的第一个对象的先决条件，依赖于第一个对象的所有对象无法退出 WAITING_ON_DEPENDENCIES 状态。

例如，请考虑以下管道定义中的摘录。在这种情况下，InputData 对象具有先决条件“Ready”，它指定在 InputData 对象完成之前，数据必须存在。如果数据不存在，InputData 对象将保持 WAITING_ON_DEPENDENCIES 状态以等待路径字段指定的数据变为可用。依赖于 InputData 的任何对象同样保持 WAITING_ON_DEPENDENCIES 状态，以等待 InputData 对象进入 FINISHED 状态。

```
{
    "id": "InputData",
    "type": "S3DataNode",
```

```
    "filePath": "s3://elasticmapreduce/samples/wordcount/wordSplitter.py",
    "schedule":{"ref":"MySchedule"},
    "precondition": "Ready"
},
{
    "id": "Ready",
    "type": "Exists"
...
```

此外，请确保您的对象拥有适当的权限，可以访问数据。在前面的示例中，如果凭证字段中的信息无权访问路径字段中指定的数据，InputData 对象将停滞在 WAITING_ON_DEPENDENCIES 状态，因为它无法访问路径字段指定的数据，即使该数据存在。

还有可能是与 Amazon S3 通信的资源没有与其关联的公有 IP 地址。例如，公有子网中的 Ec2Resource 必须具有关联的公有 IP 地址。

最后，在某些情况下，资源实例可能比计划启动关联的活动早得多地进入 WAITING_ON_DEPENDENCIES 状态，这可能会造成资源或活动失败的印象。

## 运行未按计划启动

检查您选择了正确的计划类型，确定您的任务在计划间隔的开头 (Cron 风格计划类型) 还是在计划间隔的结尾 (时间序列计划类型) 启动。

此外，检查您已在计划对象中正确指定了日期，并且 startDateTime 和 endDateTime 值使用 UTC 格式，例如在以下示例中：

```
{
    "id": "MySchedule",
    "startDateTime": "2012-11-12T19:30:00",
    "endDateTime":"2012-11-12T20:30:00",
    "period": "1 Hour",
    "type": "Schedule"
},
```

## 管道组件以错误顺序运行

您可能注意到，管道组件的开始和结束时间以错误的顺序运行，或者以与您预期不同的顺序运行。需要了解的是，如果在启动时满足了管道组件的先决条件，这些组件将同时开始运行。换句话说，不会默认按顺序执行管道组件；如果需要按特定的顺序执行，您必须使用先决条件和 dependsOn 字段控制执行顺序。

确认您使用的 dependsOn 字段填充了对相应先决条件管道组件的引用，并在组件之间具有所需的所有指针以实现所需的顺序。

## EMR 集群失败，出现错误：请求中包含的安全令牌无效

验证您的 IAM 角色、策略和信任关系，如 适用于 AWS Data Pipeline 的 IAM 角色 中所述。

## 权限不足，无法访问资源

您在 IAM 角色上设置的权限确定 AWS Data Pipeline 是否可以访问您的 EMR 集群和 EC2 实例来运行管道。此外，IAM 还提供了信任关系概念，从而允许代表您创建资源。例如，当您创建一个使用 EC2 实例运行命令来移动数据的管道时，AWS Data Pipeline 可以为您预置此 EC2 实例。如果您遇到问题，尤其是涉及到您手动可以访问但 AWS Data Pipeline 无法访问的资源，请验证您的 IAM 角色、策略和信任关系，如 适用于 AWS Data Pipeline 的 IAM 角色 中所述。

## 状态代码：400 错误代码：PipelineNotFoundException

此错误表示您的 IAM 默认角色可能没有 AWS Data Pipeline 正常工作所需的权限。有关更多信息，请参阅适用于 AWS Data Pipeline 的 IAM 角色。

## 创建管道导致安全令牌错误

在您尝试创建管道时收到了以下错误：

无法使用“pipeline_name”创建管道。错误：UnrecognizedClientException - 请求中包含的安全令牌无效。

## 在控制台中看不到管道详细信息

AWS Data Pipeline 控制台管道筛选条件应用到管道的计划 开始日期，而不考虑何时提交的管道。可以使用发生在过去的计划开始日期提交新管道，默认日期筛选条件可能不显示。要查看管道详细信息，请更改日期筛选条件，确保计划管道开始日期在日期范围筛选条件之内。

## 远程运行程序中的错误状态代码：404，Amazon Web Service：Amazon S3

此错误表示任务运行程序无法访问存储在 Amazon S3 中的您的文件。验证：

- 您已正确设置凭证
- 您尝试访问的 Amazon S3 存储桶存在
- 您有权访问 Amazon S3 存储桶

## 访问被拒绝 - 未授权执行函数 datapipeline：

在任务运行程序日志中，您可能会看到类似以下内容的错误消息：

- ERROR 状态代码：403
- Amazon Web Service：DataPipeline
- AWS 错误代码：AccessDenied
- AWS 错误消息：用户：arn:aws:sts::XXXXXXXXXXXX:federated-user/i-XXXXXXXX 未得到授权执行：datapipeline:PollForTask。

> **Note**
>
> 在该错误消息中，可能会将 PollForTask 替换为其他 AWS Data Pipeline 权限的名称。

此错误消息表示您指定的 IAM 角色需要额外的权限才能与 AWS Data Pipeline 交互。确保您的 IAM 角色策略包含以下行，其中 PollForTask 替换为您要添加的权限的名称（使用 * 以授予所有权限）。有关如何创建新 IAM 角色和将策略应用到其上的更多信息，请参阅使用 IAM 指南中的管理 IAM 策略。

```
{
"Action": [ "datapipeline:PollForTask" ],
"Effect": "Allow",
"Resource": ["*"]
}
```

## 旧 Amazon EMR AMI 可能会为大 CSV 文件创建 False 数据

在 Amazon EMR AMI 3.9（3.8 及更低版本）上，AWS Data Pipeline 使用自定义 InputFormat 读取和写入 CSV 文件用于 MapReduce 作业。这在服务对 Amazon S3 暂存表时使用。此 InputFormat 上发现问题，从大 CSV 文件上读取记录会导致生成未正确复制的表。此问题已在更高版本的 Amazon EMR 中修复。请使用 Amazon EMR AMI 3.9 或 Amazon EMR 版本 4.0.0 或更高版本。

## 提高 AWS Data Pipeline 限制

有时，您可能会超过特定 AWS Data Pipeline 系统限制。例如，默认管道限制为 20，每个管道中有 50 个对象。如果您发现自己需要的管道数超过限制，请考虑合并多个管道来创建较少数量的管道，每个管道中有更多对象。有关 AWS Data Pipeline 限制的更多信息，请参阅 AWS Data Pipeline 限制。但

是，如果您无法使用管道合并技术来解决限制问题，请使用此表单申请增加容量：提高 Data Pipeline 上限。

# AWS Data Pipeline 限制

为了确保所有用户的容量，AWS Data Pipeline 会对您可分配的资源以及资源分配速率施加限制。

目录



## 账户限制

以下限制适用于单个 AWS 账户。如果您需要额外容量，可以使用 Amazon Web Services 支持中心请求表增加容量。

| 属性 | 限制 | 可调整 |
| --- | --- | --- |
| 管线数量 | 100 | 是 |
| 每个管道的对象数量 | 100 | 是 |
| 每个对象的活动实例数量 | 5 | 是 |
| 每个对象的字段数量 | 50 | 否 |
| 每个字段名或标识符的 UTF8 字节数 | 256 | 否 |
| 每个字段的 UTF8 字节数 | 10240 | 否 |
| 每个对象的 UTF8 字节数 | 15360（包括字段名） | 否 |
| 对象的实例创建速率 | 每 5 分钟一个 | 否 |
| 管道活动的重试次数 | 每个任务 5 次 | 否 |

| 属性 | 限制 | 可调整 |
| --- | --- | --- |
| 重试之间的最短延迟 | 2 分钟 | 否 |
| 最短计划时间间隔 | 15 分钟 | 否 |
| 单个对象的累计最大数量 | 32 | 否 |
| 每个 Ec2Resource 对象的最大 EC2 实例数量 | 1 | 否 |

# Web 服务调用限制

AWS Data Pipeline 将限制可调用 Web 服务 API 的速率。这些限制也适用于代表您调用 Web 服务 API 的 AWS Data Pipeline 代理，例如控制台、CLI 和任务运行程序。

以下限制适用于单个 AWS 账户。这意味着，该账户的总使用量（包括 用户产生的使用量）不能超过这些限制。

突增速率可让您在非活动期间节省 Web 服务调用并在很短的时间内扩展这些调用。例如，CreatePipeline 的常规速率为每 5 秒执行 1 次调用。如果在 30 秒内未调用该服务，则会节省 6 次调用。您可以随后在 1 秒内调用 6 次 Web 服务。由于该值低于突增限制并确保您的平均调用次数符合常规速率限制，因此，您的调用不受限制。

如果您超出速率限制和突增限制，则您的 Web 服务调用将失败并返回限制异常。任务运行程序工作线程的默认实施自动重试发生限制异常的失败 API 调用。任务运行程序使用退避，以便后续 API 调用尝试具有逐渐增加的间隔。如果您编写工作线程，我们建议您实施类似的重试逻辑。

针对单个 AWS 账户应用这些限制。

| API | 常规速率限制 | 突增限制 |
| --- | --- | --- |
| ActivatePipeline | 每秒调用 1 次 | 100 次调用 |
| CreatePipeline | 每秒调用 1 次 | 100 次调用 |
| DeletePipeline | 每秒调用 1 次 | 100 次调用 |

| API | 常规速率限制 | 突增限制 |
| --- | --- | --- |
| DescribeObjects | 每秒调用 2 次 | 100 次调用 |
| DescribePipelines | 每秒调用 1 次 | 100 次调用 |
| GetPipelineDefinition | 每秒调用 1 次 | 100 次调用 |
| PollForTask | 每秒调用 2 次 | 100 次调用 |
| ListPipelines | 每秒调用 1 次 | 100 次调用 |
| PutPipelineDefinition | 每秒调用 1 次 | 100 次调用 |
| QueryObjects | 每秒调用 2 次 | 100 次调用 |
| ReportTaskProgress | 每秒调用 10 次 | 100 次调用 |
| SetTaskStatus | 每秒调用 10 次 | 100 次调用 |
| SetStatus | 每秒调用 1 次 | 100 次调用 |
| ReportTaskRunnerHeartbeat | 每秒调用 1 次 | 100 次调用 |
| ValidatePipelineDefinition | 每秒调用 1 次 | 100 次调用 |

# 扩展注意事项

AWS Data Pipeline 扩展以满足大量并发任务，您可以将其配置为自动创建所需的资源以处理大型工作负载。这些自动创建的资源由您控制，并计入您的 AWS 账户资源限制。例如，如果您配置 AWS Data Pipeline 自动创建 20 个节点的 Amazon EMR 集群以处理数据，并且您的 AWS 账户的 EC2 实例限制设置为 20，您可能会无意中用尽可用的回填资源。因此，在设计中请考虑这些资源限制或相应增加您的账户限制。

如果您需要额外容量，可以使用 Amazon Web Services 支持中心请求表增加容量。

# AWS Data Pipeline 资源

以下资源可帮助您使用 AWS Data Pipeline。

- AWS Data Pipeline 产品信息 –提供 AWS Data Pipeline 相关信息的主要网页。
- AWS Data Pipeline 技术常见问题 –涵盖了开发人员对此产品提出的 20 个最热门的问题。
- 发行说明 –概括介绍了当前发布版本的情况。特别说明了一些新功能、修复和已知问题。
- AWS Data Pipeline 开发论坛 – 社区形式的论坛，开发人员在此讨论与 Amazon Web Services 有关的技术问题。

- 课程和研讨会 – 指向基于角色的专业课程和自主进度动手实验室的链接，这些课程和实验室旨在帮助您增强 AWS 技能并获得实践经验。
- AWS 开发人员中心 – 浏览教程、下载工具并了解 AWS 开发人员活动。
- AWS 开发人员工具 – 指向开发人员工具、开发工具包、IDE 工具包和命令行工具的链接，这些资源用于开发和管理 AWS 应用程序。
- 入门资源中心 – 了解如何设置 AWS 账户、加入 AWS 社区和启动您的第一个应用程序。
- 动手实践教程 – 按照分步教程在 AWS 上启动您的第一个应用程序。
- AWS 白皮书 – 指向 AWS 技术白皮书的完整列表的链接，这些资料涵盖了架构、安全性、经济性等主题，由 AWS 解决方案架构师或其他技术专家编写。
- AWS Support 中心 – 用于创建和管理 AWS Support 案例的中心。还提供指向其他有用资源的链接，如论坛、技术常见问题、服务运行状况以及AWS Trusted Advisor。
- AWS Support – 提供有关 AWS Support 的信息的主要网页，这是一个一对一的快速响应支持渠道，可以帮助您在云中构建和运行应用程序。
- 联系我们 – 用于查询有关AWS账单、账户、事件、滥用和其他问题的中央联系点。
- AWS 网站条款 – 有关我们的版权和商标、您的账户、许可、网站访问和其他主题的详细信息。

# 文档历史记录

本文档与 AWS Data Pipeline 的 2012 年 10 月 29 日版本关联。

| 更改 | 描述 | 发行日期 |
| --- | --- | --- |
| 添加了有关使用 AWS CLI 执行某些过程的文档。删除了与 AWS Data Pipeline 控制台相关的过程。 | 有关更多信息，请参阅克隆管道、查看管道日志和使用 CLI 从 Data Pipeline 模板创建管道。 | 2023 年 5 月 26 日 |
| 添加了更多内容和从 AWS Data Pipeline 迁移到其他替代服务的示例。 | 更新了关于迁移 AWS Data Pipeline 到 AWS Glue、AWS Step Functions 或 Amazon MWAA 的主题，提供了有关每种替代方案、服务之间的概念映射以及示例的更多信息。有关更多信息，请参阅从 AWS Data Pipeline 迁移工作负载。 | 2023 年 3 月 31 日 |
| 添加了有关 AWS Data Pipeline 支持 IMDSv2 的信息。 | AWS Data Pipeline 支持 Amazon EMR 的 IMDSv2 以及 Amazon EC2 资源。有关更多信息，请参阅AWS Data Pipeline 中的数据保护、EmrCluster和Ec2Resource。 | 2022 年 12 月 16 日 |
| 添加了有关从 AWS Data Pipeline 迁移到其他替代服务的主题。 | 现在还有其他 AWS 服务可以为客户提供更好的数据集成体验。您可以将 AWS Data Pipeline 的典型用例迁移到 AWS Glue、AWS Step Functions 或 Amazon MWAA。有关更多信息，请参阅从 AWS Data Pipeline 迁移工作负载。 | 2022 年 12 月 16 日 |
| 更新了支持的 Amazon EC2 和 Amazon EMR 实例列表。 | 更新了支持的 Amazon EC2 和 Amazon EMR 实例列表。有关更多信息，请参阅管道工作活动支持的实例类型。<br><br>更新了用于实例的 HVM（硬件虚拟机）AMI 的 ID 列表。有关更多信息，请参阅语法并搜索 `imageId`。 | 2018 年 11 月 9 日 |

| 更改 | 描述 | 发行日期 |
| --- | --- | --- |
| 更新了用于实例的 HVM（硬件虚拟机）AMI 的 ID 列表。 | | |
| 添加了将 Amazon EBS 卷附加到集群节点以及在私有子网中启动 Amazon EMR 集群的配置。 | 在 `EMRcluster` 对象中添加了配置选项。您可以在使用 Amazon EMR 集群的管道中使用这些选项。<br><br>请使用 `coreEbsConfiguration`、`masterEbsConfiguration` 和 `TaskEbsConfiguration` 字段配置将 Amazon EBS 卷附加到 Amazon EMR 集群中的核心节点、主节点和任务节点的操作。有关更多信息，请参阅将 EBS 卷附加到集群节点。<br><br>可以使用 `emrManagedMasterSecurityGroupId`、`emrManagedSlaveSecurityGroupId` 和 `ServiceAccessSecurityGroupId` 字段在私有子网中配置 Amazon EMR 集群。有关更多信息，请参阅在私有子网中配置 Amazon EMR 集群。<br><br>有关 `EMRcluster` 语法的更多信息，请参阅EmrCluster。 | 2018 年 4 月 19 日 |
| 添加了支持的 Amazon EC2 和 Amazon EMR 实例列表。 | 增加了在管道定义中未指定实例类型时 AWS Data Pipeline 默认创建的实例列表。添加了支持的 Amazon EC2 和 Amazon EMR 实例列表。有关更多信息，请参阅管道工作活动支持的实例类型。 | 2018 年 3 月 22 日 |
| 增加了对按需管道的支持。 | • 增加了对按需管道的支持，使您能够通过重新激活管道来重新运行它。 | 2016 年 2 月 22 日 |
| 对 RDS 数据库的其他支持 | • 已将 `rdsInstanceId`、`region` 和 `jdbcDriverJarUri` 添加到 RdsDatabase。<br>• 更新了 SqlActivity 中的 `database` 以支持 `RdsDatabase`。 | 2015 年 8 月 17 日 |

| 更改 | 描述 | 发行日期 |
| --- | --- | --- |
| 其他 JDBC 支持 | • 更新了 SqlActivity 中的 database 以支持 JdbcDatabase 。<br>• 已将 jdbcDriverJarUri 添加到 JdbcDatabase<br>• 已将 initTimeout 添加到 Ec2Resource 和 EmrCluster。<br>• 已将 runAsUser 添加到 Ec2Resource。 | 2015 年 7 月 7 日 |
| HadoopActivity、可用区和 Spot 实例支持 | • 增加了对将并行工作提交到 Hadoop 集群的支持。有关更多信息，请参阅HadoopActivity。<br>• 增加了使用 Ec2Resource 和 EmrCluster 请求 Spot 实例的功能。<br>• 增加了在指定的可用区中启动 EmrCluster 资源的功能。 | 2015 年 6 月 1 日 |
| 停用管道 | 增加了对停用活动管道的支持。有关更多信息，请参阅停用管道。 | 2015 年 4 月 7 日 |
| 更新了模板和控制台 | 添加了新模板。更新了“入门”一章以使用 Getting Started with ShellCommandActivity 模板。有关更多信息，请参阅使用 CLI 从 Data Pipeline 模板创建管道。 | 2014 年 11 月 25 日 |
| VPC 支持 | 增加了对在 Virtual Private Cloud (VPC) 中启动资源的支持。 | 2014 年 3 月 12 日 |
| 区域支持 | 增加了对多个服务区域的支持。除了 us-east-1 之外，AWS Data Pipeline 也在 eu-west-1 、ap-northeast-1 、ap-southeast-2 和 us-west-2 中受支持。 | 2014 年 2 月 20 日 |

| 更改 | 描述 | 发行日期 |
| --- | --- | --- |
| Amazon Redshift 支持 | 在 AWS Data Pipeline 中增加了对 Amazon Redshift 的支持，包括一个新的控制台模板 (Copy to Redshift) 和一个模板演示教程。有关更多信息，请参阅 使用 AWS Data Pipeline 复制数据到 Amazon Redshift、RedshiftDataNode、RedshiftDatabase 和 RedshiftCopyActivity。 | 2013 年 11 月 6 日 |
| PigActivity | 增加了 PigActivity，它提供了对 Pig 的本机支持。有关更多信息，请参阅PigActivity。 | 2013 年 10 月 15 日 |
| 新的控制台模板、活动和数据格式 | 增加了新的 CrossRegion DynamoDB Copy 控制台模板，包括新的 HiveCopyActivity 和 DynamoDBExportDataFormat。 | 2013 年 8 月 21 日 |
| 级联故障和重新运行 | 增加了有关 AWS Data Pipeline 级联故障和重新运行行为的信息。有关更多信息，请参阅级联故障和重新运行。 | 2013 年 8 月 8 日 |
| 问题排查视频 | 增加了 AWS Data Pipeline 基本问题排查视频。有关更多信息，请参阅故障排除。 | 2013 年 7 月 17 日 |
| 编辑活动管道 | 添加了有关编辑活动管道和重新运行管道组件的更多信息。有关更多信息，请参阅编辑管道。 | 2013 年 7 月 17 日 |
| 使用不同区域中的资源 | 添加了有关使用不同区域中的资源的更多信息。有关更多信息，请参阅利用多个区域中的资源使用管道。 | 2013 年 6 月 17 日 |
| WAITING_ON_DEPENDENCIES 状态 | CHECKING_PRECONDITIONS 状态已更改为 WAITING_ON_DEPENDENCIES，并且已为管道对象添加 @waitingOn 运行时字段。 | 2013 年 5 月 20 日 |
| DynamoDBDataFormat | 添加了 DynamoDBDataFormat 模板。 | 2013 年 4 月 23 日 |
| 处理 Web 日志视频和 Spot 实例支持 | 推出了“使用 AWS Data Pipeline、Amazon EMR 和 Hive 处理 Web 日志”视频以及 Amazon EC2 竞价型实例支持。 | 2013 年 2 月 21 日 |

| 更改 | 描述 | 发行日期 |
| --- | --- | --- |
| | 首次发布 AWS Data Pipeline 开发人员指南。 | 2012 年 12 月 20 日 |