AWS Data Pipeline

Developer Guide

API Version 2012-10-29

AWS Data Pipeline: Developer Guide

Table of Contents

What is AWS Data Pipeline?

AWS Data Pipeline is a web service that you can use to automate the movement and transformation of data. With AWS Data Pipeline, you can define data-driven workflows, so that tasks can be dependent on the successful completion of previous tasks. You define the parameters of your data transformations and AWS Data Pipeline enforces the logic that you've set up.

The following components of AWS Data Pipeline work together to manage your data:

- A *pipeline definition* specifies the business logic of your data management. For more information, see Pipeline Definition File Syntax (p. 54).
- A *pipeline* schedules and runs tasks. You upload your pipeline definition to the pipeline, and then activate the pipeline. You can edit the pipeline definition for a running pipeline and activate the pipeline again for it to take effect. You can deactivate the pipeline, modify a data source, and then activate the pipeline again. When you are finished with your pipeline, you can delete it.
- *Task Runner* polls for tasks and then performs those tasks. For example, Task Runner could copy log files to Amazon S3 and launch Amazon EMR clusters. Task Runner is installed and runs automatically on resources created by your pipeline definitions. You can write a custom task runner application, or you can use the Task Runner application that is provided by AWS Data Pipeline. For more information, see Task Runners (p. 5).

For example, you can use AWS Data Pipeline to archive your web server's logs to Amazon Simple Storage Service (Amazon S3) each day and then run a weekly Amazon EMR (Amazon EMR) cluster over those logs to generate traffic reports. AWS Data Pipeline schedules the daily tasks to copy data and the weekly task to launch the Amazon EMR cluster. AWS Data Pipeline also ensures that Amazon EMR waits for the final day's data to be uploaded to Amazon S3 before it begins its analysis, even if there is an unforeseen delay in uploading the logs.

Related Services

AWS Data Pipeline works with the following services to store data.

- Amazon DynamoDB — Provides a fully-managed NoSQL database with fast performance at a low cost. For more information, see *Amazon DynamoDB Developer Guide*.
- Amazon RDS — Provides a fully-managed relational database that scales to large datasets. For more information, see *Amazon Relational Database Service Developer Guide*.

- Amazon Redshift — Provides a fast, fully-managed, petabyte-scale data warehouse that makes it easy and cost-effective to analyze a vast amount of data. For more information, see *Amazon Redshift Database Developer Guide*.
- Amazon S3 — Provides secure, durable, and highly-scalable object storage. For more information, see *Amazon Simple Storage Service Developer Guide*.

AWS Data Pipeline works with the following compute services to transform data.

- Amazon EC2 — Provides resizeable computing capacity—literally, servers in Amazon's data centers —that you use to build and host your software systems. For more information, see *Amazon EC2 User Guide for Linux Instances*.
- Amazon EMR — Makes it easy, fast, and cost-effective for you to distribute and process vast amounts of data across Amazon EC2 servers, using a framework such as Apache Hadoop or Apache Spark. For more information, see *Amazon EMR Developer Guide*.

Accessing AWS Data Pipeline

You can create, access, and manage your pipelines using any of the following interfaces:

- **AWS Management Console**— Provides a web interface that you can use to access AWS Data Pipeline.
- **AWS Command Line Interface (AWS CLI)** — Provides commands for a broad set of AWS services, including AWS Data Pipeline, and is supported on Windows, Mac, and Linux. For more information about installing the AWS CLI, see AWS Command Line Interface. For a list of commands for AWS Data Pipeline, see datapipeline.
- **AWS SDKs** — Provides language-specific APIs and takes care of many of the connection details, such as calculating signatures, handling request retries, and error handling. For more information, see AWS SDKs.
- **Query API**— Provides low-level APIs that you call using HTTPS requests. Using the Query API is the most direct way to access AWS Data Pipeline, but it requires that your application handle low-level details such as generating the hash to sign the request, and error handling. For more information, see the *AWS Data Pipeline API Reference*.

Pricing

With Amazon Web Services, you pay only for what you use. For AWS Data Pipeline, you pay for your pipeline based on how often your activities and preconditions are scheduled to run and where they run. For pricing information, see AWS Data Pipeline Pricing

If your AWS account is less than 12 months old, you are eligible to use the free tier. The free tier includes 3 low-frequency preconditions and 5 low-frequency activities per month at no charge. For more information, see AWS Free Tier.

Data Pipeline Concepts

Before you begin, read about the key concepts and components for AWS Data Pipeline.

Contents

Pipeline Definition

A pipeline definition is how you communicate your business logic to AWS Data Pipeline. It contains the following information:

- Names, locations, and formats of your data sources
- Activities that transform the data
- The schedule for those activities
- Resources that run your activities and preconditions
- Preconditions that must be satisfied before the activities can be scheduled
- Ways to alert you with status updates as pipeline execution proceeds

From your pipeline definition, AWS Data Pipeline determines the tasks that will occur, schedules them, and assigns them to task runners. If a task is not completed successfully, AWS Data Pipeline retries the task according to your instructions and, if necessary, reassigns it to another task runner. If the task fails repeatedly, you can configure the pipeline to notify you.

For example, in your pipeline definition, you might specify that log files generated by your application are archived each month in 2013 to an Amazon S3 bucket. AWS Data Pipeline would then create 12 tasks,

each copying over a month's worth of data, regardless of whether the month contained 30, 31, 28, or 29 days.

You can create a pipeline definition in the following ways:

- Graphically, by using the AWS Data Pipeline console
- Textually, by writing a JSON file in the format used by the command line interface
- Programmatically, by calling the web service with either one of the AWS SDKs or the AWS Data Pipeline API

A pipeline definition can contain the following types of components.

Pipeline Components

Data Nodes (p. 5)

The location of input data for a task or the location where output data is to be stored.

Activities (p. 6)

A definition of work to perform on a schedule using a computational resource and typically input and output data nodes.

Preconditions (p. 7)

A conditional statement that must be true before an action can run.

Scheduling Pipelines (p. 15)

Defines the timing of a scheduled event, such as when an activity runs.

Resources (p. 7)

The computational resource that performs the work that a pipeline defines.

Actions (p. 9)

An action that is triggered when specified conditions are met, such as the failure of an activity.

For more information, see Pipeline Definition File Syntax (p. 54).

Pipeline Components, Instances, and Attempts

There are three types of items associated with a scheduled pipeline:

- **Pipeline Components** — Pipeline components represent the business logic of the pipeline and are represented by the different sections of a pipeline definition. Pipeline components specify the data sources, activities, schedule, and preconditions of the workflow. They can inherit properties from parent components. Relationships among components are defined by reference. Pipeline components define the rules of data management; they are not a to-do list.
- **Instances** — When AWS Data Pipeline runs a pipeline, it compiles the pipeline components to create a set of actionable instances. Each instance contains all the information needed to perform a specific task. The complete set of instances is the to-do list of the pipeline. AWS Data Pipeline hands the instances out to task runners to process.
- **Attempts** — To provide robust data management, AWS Data Pipeline retries a failed operation. It continues to do so until the task reaches the maximum number of allowed retry attempts. Attempt objects track the various attempts, results, and failure reasons if applicable. Essentially, it is the instance with a

counter. AWS Data Pipeline performs retries using the same resources from the previous attempts, such as Amazon EMR clusters and EC2 instances.

Note
Retrying failed tasks is an important part of a fault tolerance strategy, and AWS Data Pipeline pipeline definitions provide conditions and thresholds to control retries. However, too many retries can delay detection of an unrecoverable failure because AWS Data Pipeline does not report failure until it has exhausted all the retries that you specify. The extra retries may accrue additional charges if they are running on AWS resources. As a result, carefully consider when it is appropriate to exceed the AWS Data Pipeline default settings that you use to control re-tries and related settings.

Task Runners

A task runner is an application that polls AWS Data Pipeline for tasks and then performs those tasks.

Task Runner is a default implementation of a task runner that is provided by AWS Data Pipeline. When Task Runner is installed and configured, it polls AWS Data Pipeline for tasks associated with pipelines that you have activated. When a task is assigned to Task Runner, it performs that task and reports its status back to AWS Data Pipeline.

The following diagram illustrates how AWS Data Pipeline and a task runner interact to process a scheduled task. A task is a discrete unit of work that the AWS Data Pipeline service shares with a task runner and differs from a pipeline, which is a general definition of activities and resources that usually yields several tasks.

There are two ways you can use Task Runner to process your pipeline:

- AWS Data Pipeline installs Task Runner for you on resources that are launched and managed by the AWS Data Pipeline web service.
- You install Task Runner on a computational resource that you manage, such as a long-running EC2 instance, or an on-premise server.

For more information about working with Task Runner, see Working with Task Runner (p. 263).

Data Nodes

In AWS Data Pipeline, a data node defines the location and type of data that a pipeline activity uses as input or output. AWS Data Pipeline supports the following types of data nodes:

DynamoDBDataNode (p. 131)

An DynamoDB table that contains data for HiveActivity (p. 168) or EmrActivity (p. 156) to use.

SqlDataNode (p. 147)

A SQL table and database query that represents data for a pipeline activity to use.

Note
Previously, MySqlDataNode was used but that is a deprecated data node. Please use SqlDataNode instead.

RedshiftDataNode (p. 139)

An Amazon Redshift table that contains data for RedshiftCopyActivity (p. 187) to use.

S3DataNode (p. 143)

An Amazon S3 location that contains one or more files for a pipeline activity to use.

Databases

AWS Data Pipeline supports the following types of databases:

JdbcDatabase (p. 237)

A JDBC database.

RdsDatabase (p. 239)

An Amazon RDS database.

RedshiftDatabase (p. 240)

An Amazon Redshift database.

Activities

In AWS Data Pipeline, an activity is a pipeline component that defines the work to perform. AWS Data Pipeline provides several pre-packaged activities that accommodate common scenarios, such as moving data from one location to another, running Hive queries, and so on. Activities are extensible, so you can run your own custom scripts to support endless combinations.

AWS Data Pipeline supports the following types of activities:

CopyActivity (p. 152)

Copies data from one location to another.

EmrActivity (p. 156)

Runs an Amazon EMR cluster.

HiveActivity (p. 168)

Runs a Hive query on an Amazon EMR cluster.

HiveCopyActivity (p. 174)

Runs a Hive query on an Amazon EMR cluster with support for advanced data filtering and support for S3DataNode (p. 143) and DynamoDBDataNode (p. 131).

PigActivity (p. 179)

Runs a Pig script on an Amazon EMR cluster.

RedshiftCopyActivity (p. 187)

Copies data to and from Amazon Redshift tables.

ShellCommandActivity (p. 194)

Runs a custom UNIX/Linux shell command as an activity.

SqlActivity (p. 199)

Runs a SQL query on a database.

Some activities have special support for staging data and database tables. For more information, see Staging Data and Tables with Pipeline Activities (p. 45).

Preconditions

In AWS Data Pipeline, a precondition is a pipeline component containing conditional statements that must be true before an activity can run. For example, a precondition can check whether source data is present before a pipeline activity attempts to copy it. AWS Data Pipeline provides several pre-packaged preconditions that accommodate common scenarios, such as whether a database table exists, whether an Amazon S3 key is present, and so on. However, preconditions are extensible and allow you to run your own custom scripts to support endless combinations.

There are two types of preconditions: system-managed preconditions and user-managed preconditions. System-managed preconditions are run by the AWS Data Pipeline web service on your behalf and do not require a computational resource. User-managed preconditions only run on the computational resource that you specify using the `runsOn` or `workerGroup` fields. The `workerGroup` resource is derived from the activity that uses the precondition.

System-Managed Preconditions

DynamoDBDataExists (p. 223)

Checks whether data exists in a specific DynamoDB table.

DynamoDBTableExists (p. 225)

Checks whether a DynamoDB table exists.

S3KeyExists (p. 230)

Checks whether an Amazon S3 key exists.

S3PrefixNotEmpty (p. 232)

Checks whether an Amazon S3 prefix is empty.

User-Managed Preconditions

Exists (p. 227)

Checks whether a data node exists.

ShellCommandPrecondition (p. 234)

Runs a custom Unix/Linux shell command as a precondition.

Resources

In AWS Data Pipeline, a resource is the computational resource that performs the work that a pipeline activity specifies. AWS Data Pipeline supports the following types of resources:

Ec2Resource (p. 204)

An EC2 instance that performs the work defined by a pipeline activity.

EmrCluster (p. 210)

An Amazon EMR cluster that performs the work defined by a pipeline activity, such as EmrActivity (p. 156).

Resources can run in the same region with their working data set, even a region different than AWS Data Pipeline. For more information, see Using a Pipeline with Resources in Multiple Regions (p. 51).

Resource Limits

AWS Data Pipeline scales to accommodate a huge number of concurrent tasks and you can configure it to automatically create the resources necessary to handle large workloads. These automatically-created resources are under your control and count against your AWS account resource limits. For example, if you configure AWS Data Pipeline to create a 20-node Amazon EMR cluster automatically to process data and your AWS account has an EC2 instance limit set to 20, you may inadvertently exhaust your available backfill resources. As a result, consider these resource restrictions in your design or increase your account limits accordingly. For more information about service limits, see AWS Service Limits in the *AWS General Reference*.

Note
The limit is 1 instance per `Ec2Resource` component object

Supported Platforms

Pipelines can launch your resources into the following platforms:

EC2-Classic

Your resources run in a single, flat network that you share with other customers.

EC2-VPC

Your resources run in a virtual private cloud (VPC) that's logically isolated to your AWS account.

Your AWS account is capable of launching resources either into both platforms or only into EC2-VPC, on a region by region basis. For more information, see Supported Platforms in the *Amazon EC2 User Guide for Linux Instances*.

If your AWS account supports only EC2-VPC, we create a default VPC for you in each AWS region. By default, we launch your resources into a default subnet of your default VPC. Alternatively, you can create a nondefault VPC and specify one of its subnets when you configure your resources, and then we'll launch your resources into the specified subnet of the nondefault VPC.

When you launch an instance into a VPC, you must specify a security group created specifically for that VPC. You can't specify a security group that you created for EC2-Classic when you launch an instance into a VPC. In addition, you must use the security group ID and not the security group name to identify a security group for a VPC.

For more information about using a VPC with AWS Data Pipeline, see Launching Resources for Your Pipeline into a VPC (p. 48).

Amazon EC2 Spot Instances with Amazon EMR Clusters and AWS Data Pipeline

Pipelines can use Amazon EC2 Spot Instances for the task nodes in their Amazon EMR cluster resources. By default, pipelines use on-demand EC2 instances. Spot Instances let you bid on spare EC2 instances

and run them whenever your bid exceeds the current Spot Price, which varies in real-time based on supply and demand. The Spot Instance pricing model complements the on-demand and Reserved Instance pricing models, potentially providing the most cost-effective option for obtaining compute capacity, depending on your application. For more information, see the Amazon EC2 Spot Instances product page.

When you use Spot Instances, AWS Data Pipeline submits your Spot Instance bid to Amazon EMR when your cluster is launched. After your bid succeeds, Amazon EMR automatically allocates the cluster's work to the number of Spot Instance task nodes that you define using the `taskInstanceCount` field. AWS Data Pipeline limits Spot Instances for task nodes to ensure that on-demand core nodes are available to run your pipeline if you don't successfully bid on a Spot Instance.

You can edit a failed or completed pipeline resource instance to add Spot Instances; when the pipeline re-launches the cluster, it uses Spot Instances for the task nodes.

Spot Instances Considerations

When you use Spot Instances with AWS Data Pipeline, the following considerations apply:

- Spot Instances can terminate at any time if you lose the bid. However, you do not lose your data because AWS Data Pipeline employs clusters with core nodes that are always on-demand instances and not subject to bid-related termination.
- Spot Instances can take more time to start due to the bidding and termination process; therefore, a Spot Instance-based pipeline could run more slowly than an equivalent on-demand instance pipeline.
- Your cluster might not run if you do not receive your Spot Instances, such as when your bid price is too low. For more information, see Troubleshooting Spot Instances in the *Amazon EMR Developer Guide*.

Actions

AWS Data Pipeline actions are steps that a pipeline component takes when certain events occur, such as success, failure, or late activities. The event field of an activity refers to an action, such as a reference to `Terminate` in the `onLateAction` field of `EmrActivity`. AWS Data Pipeline supports the following actions:

SnsAlarm (p. 250)

An action that sends an Amazon SNS notification to a topic based on certain events.

Terminate (p. 251)

An action that triggers the cancellation of a pending or unfinished activity, resource, or data node.

Even though the AWS Data Pipeline console and CLI convey pipeline status information, AWS Data Pipeline relies on Amazon SNS notifications as the primary way to indicate the status of pipelines and their components in an unattended manner. For more information, see Amazon Simple Notification Service (Amazon SNS).

Proactively Monitoring Pipelines

The best way to detect problems is to monitor your pipelines proactively from the start. You can configure pipeline components to inform you of certain situations or events, such as when a pipeline component fails or doesn't begin by its scheduled start time. AWS Data Pipeline makes it easy to configure notifications by providing event fields on pipeline components that you can associate with Amazon SNS notifications, such as `onSuccess`, `OnFail`, and `onLateAction`.

Setting Up for AWS Data Pipeline

Before you use AWS Data Pipeline for the first time, complete the following tasks.

Tasks

- Sign Up for AWS (p. 10)
- Create the Required IAM Roles (for CLI or API only) (p. 10)

After you complete these tasks, you can start using AWS Data Pipeline. For a basic tutorial, see Getting Started with AWS Data Pipeline (p. 12).

Sign Up for AWS

When you sign up for Amazon Web Services (AWS), your AWS account is automatically signed up for all services in AWS, including AWS Data Pipeline. You are charged only for the services that you use. For more information about AWS Data Pipeline usage rates, see AWS Data Pipeline.

If you have an AWS account already, skip to the next task. If you don't have an AWS account, use the following procedure to create one.

To create an AWS account

1. Open https://aws.amazon.com/, and then choose **Create an AWS Account**.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.

Create the Required IAM Roles (for CLI or API only)

AWS Data Pipeline requires IAM roles to determine what actions your pipelines can perform and what resources it can access. Additionally, when your pipeline creates a resource, such as an EC2 instance or EMR cluster, IAM roles determine what actions your applications can perform and what resources they can access.

The AWS Data Pipeline console creates the following roles for you:

- **DataPipelineDefaultRole** - Grants AWS Data Pipeline access to your AWS resources
- **DataPipelineDefaultResourceRole** - Grants the applications on your EC2 instances access to your AWS resources

If you have used AWS Data Pipeline previously and have existing versions of these IAM roles, you might need to update them. For more information, see Update Existing IAM Roles for AWS Data Pipeline (p. 66).

If you are using a CLI or an API and you have not used the AWS Data Pipeline console to create a pipeline previously, you must create these roles manually using AWS Identity and Access Management (IAM).

To manually create the required IAM roles

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. Create the **DataPipelineDefaultRole** role as follows:
   a. In the navigation pane, click **Roles**, and then click **Create New Role**.
   b. On the **Set Role Name** page, enter `DataPipelineDefaultRole` as the role name.
   c. On the **Select Role Type** page, under **AWS Service Roles**, click **Select** in the row for **AWS Data Pipeline**.
   d. On the **Attach Policy** page, click the box next to the **AWSDataPipelineRole** policy, and then click **Next Step**.
   e. On the **Review** page, click **Create Role**.
3. Create the **DataPipelineDefaultResourceRole** role as follows:
   a. In the navigation pane, click **Roles**, and then click **Create New Role**.
   b. On the **Set Role Name** page, enter `DataPipelineDefaultResourceRole` as the role name.
   c. On the **Select Role Type** page, under **AWS Service Roles**, click **Select** in the row for **Amazon EC2 Role for Data Pipeline**.
   d. On the **Attach Policy** page, click the box next to the **AmazonEC2RoleforDataPipelineRole** policy, and then click **Next Step**.
   e. On the **Review** page, click **Create Role**.

Alternatively, you can create and use custom roles. For an example of how to specify custom roles for an `EmrCluster` object, see Specify custom IAM roles (p. 212).

Getting Started with AWS Data Pipeline

AWS Data Pipeline helps you sequence, schedule, run, and manage recurring data processing workloads reliably and cost-effectively. This service makes it easy for you to design extract-transform-load (ETL) activities using structured and unstructured data, both on-premises and in the cloud, based on your business logic.

To use AWS Data Pipeline, you create a *pipeline definition* that specifies the business logic for your data processing. A typical pipeline definition consists of activities (p. 6) that define the work to perform, data nodes (p. 5) that define the location and type of input and output data, and a schedule (p. 15) that determines when the activities are performed.

In this tutorial, you run a shell command script that counts the number of GET requests in Apache web server logs. This pipeline runs every 15 minutes for an hour, and writes output to Amazon S3 on each iteration.

Prerequisites

Before you begin, complete the tasks in Setting Up for AWS Data Pipeline (p. 10).

Pipeline Objects

The pipeline uses the following objects:

ShellCommandActivity (p. 194)

Reads the input log file and counts the number of errors.

S3DataNode (p. 143) (input)

The S3 bucket that contains the input log file.

S3DataNode (p. 143) (output)

The S3 bucket for the output.

Ec2Resource (p. 204)

The compute resource that AWS Data Pipeline uses to perform the activity.

Note that if you have a large amount of log file data, you can configure your pipeline to use an EMR cluster to process the files instead of an EC2 instance.

Schedule (p. 252)

Defines that the activity is performed every 15 minutes for an hour.

Tasks

- Create the Pipeline (p. 13)
- Monitor the Running Pipeline (p. 13)
- View the Output (p. 14)
- Delete the Pipeline (p. 14)

Create the Pipeline

The quickest way to get started with AWS Data Pipeline is to use a pipeline definition called a *template*.

To create the pipeline

1. Open the AWS Data Pipeline console at https://console.aws.amazon.com/datapipeline/.
2. From the navigation bar, select a region. You can select any region that's available to you, regardless of your location. Many AWS resources are specific to a region, but AWS Data Pipeline enables you to use resources that are in a different region than the pipeline.
3. The first screen that you see depends on whether you've created a pipeline in the current region.
   a. If you haven't created a pipeline in this region, the console displays an introductory screen. Choose **Get started now**.
   b. If you've already created a pipeline in this region, the console displays a page that lists your pipelines for the region. Choose **Create new pipeline**.
4. In **Name**, enter a name for your pipeline.
5. (Optional) In **Description**, enter a description for your pipeline.
6. For **Source**, select **Build using a template**, and then select the following template: **Getting Started using ShellCommandActivity**.
7. Under the **Parameters** section, which opened when you selected the template, leave **S3 input folder** and **Shell command to run** with their default values. Click the folder icon next to **S3 output folder**, select one of your buckets or folders, and then click **Select**.
8. Under **Schedule**, leave the default values. When you activate the pipeline the pipeline runs start, and then continue every 15 minutes for an hour.

   If you prefer, you can select **Run once on pipeline activation** instead.
9. Under **Pipeline Configuration**, leave logging enabled. Choose the folder icon under **S3 location for logs**, select one of your buckets or folders, and then choose **Select**.

   If you prefer, you can disable logging instead.
10. Under **Security/Access**, leave **IAM roles** set to **Default**.
11. Click **Activate**.

    If you prefer, you can choose **Edit in Architect** to modify this pipeline. For example, you can add preconditions.

Monitor the Running Pipeline

After you activate your pipeline, you are taken to the **Execution details** page where you can monitor the progress of your pipeline.

To monitor the progress of your pipeline

1. Click **Update** or press F5 to update the status displayed.

   Tip
   If there are no runs listed, ensure that **Start (in UTC)** and **End (in UTC)** cover the scheduled start and end of your pipeline, and then click **Update**.
2. When the status of every object in your pipeline is `FINISHED`, your pipeline has successfully completed the scheduled tasks.
3. If your pipeline doesn't complete successfully, check your pipeline settings for issues. For more information about troubleshooting failed or incomplete instance runs of your pipeline, see Resolving Common Problems (p. 273).

View the Output

Open the Amazon S3 console and navigate to your bucket. If you ran your pipeline every 15 minutes for an hour, you'll see four time-stamped subfolders. Each subfolder contains output in a file named `output.txt`. Because we ran the script on the same input file each time, the output files are identical.

Delete the Pipeline

To stop incurring charges, delete your pipeline. Deleting your pipeline deletes the pipeline definition and all associated objects.

To delete your pipeline

1. On the **List Pipelines** page, select your pipeline.
2. Click **Actions**, and then choose **Delete**.
3. When prompted for confirmation, choose **Delete**.

If you are finished with the output from this tutorial, delete the output folders from your Amazon S3 bucket.

Working with Pipelines

You can administer, create, and modify pipelines using the AWS Data Pipeline console, an AWS SDK, or the command line interface (CLI). The following sections introduce fundamental AWS Data Pipeline concepts and show you how to work with pipelines.

Important
Before you begin, see Setting Up for AWS Data Pipeline (p. 10).

Contents

Scheduling Pipelines

In AWS Data Pipeline, a schedule defines the timing of a scheduled event, such as when an activity runs. AWS Data Pipeline exposes this functionality through the Schedule (p. 252) pipeline component.

Creating a Schedule Using the Console

The AWS Data Pipeline console allows you to schedule and create pipelines. This is useful for testing and prototyping pipelines before establishing them for production workloads.

The **Create Pipeline** section has the following fields:

Field	Action
Name	Enter a name for the pipeline.
Description	(Optional) Enter a description for the pipeline.

The **Schedule** section has the following fields:

Field	Action
Run	• Choose **on activation** to run the pipeline as an on-demand pipeline. This will create a pipeline that can be run when it is activated.
Run every	Enter a period for every pipeline run.
Starting	Enter a time and date for the pipeline to start. Alternatively, your start date and time are automatically selected at pipeline activation.
Ending	Enter a time and date for the pipeline to end. If you select **never**, your pipeline continues to execute indefinitely.

The **IAM Roles & Permissions** section has the following options:

Field	Action
Default	Choose this to have AWS Data Pipeline determine the roles for you.
Custom	Choose this to designate your own IAM roles. If you select this option, you can choose the following roles: • Pipeline role—the role that determines what AWS Data Pipeline can do with resources in the account. • EC2 instance role—the role that controls what Amazon EC2 applications can do with resources in the account.

On-demand

Note
You can find the Default object on the Architect page in the **Other** section.

AWS Data Pipeline offers an on-demand schedule type, which gives the option for a pipeline to be run on pipeline activation. The pipeline will be run once in response to an activation request.

On-demand pipelines only require the schedule type to be set to `ondemand` on the default object. On-demand pipelines require that you *not* use a schedule object and they do not allow for multiple schedules. The maximum number of concurrent executions of an on-demand pipeline can be configured using the slot maxActiveInstances in the Default object. The default value for this slot is 1 for on-demand pipelines and can have a maximum value of 5.

The following Default object will use on-demand scheduling:

```
{
  "name": "Default",
  "resourceRole": "DataPipelineDefaultResourceRole",
  "role": "DataPipelineDefaultRole",
  "scheduleType": "ondemand"
```

```
}
```

Time Series Style vs. Cron Style

AWS Data Pipeline offers two types of periodic pipeline component scheduling: time series style scheduling and cron style scheduling. The schedule type allows you to specify whether the pipeline component instances should start at the beginning of the interval (also known as the period) or at the end of the interval. Time series style scheduling means instances are scheduled at the end of each interval and cron style scheduling means instances are scheduled at the beginning of each interval. For example, using time series style scheduling, if the start time is 22:00 UTC and the interval/period is set to 30 minutes, then the pipeline component instance's first run starts at 22:30 UTC, not 22:00 UTC. If you want the instance to run at the beginning of the period/interval, such as 22:00 UTC, use cron style scheduling instead.

Note
The minimum scheduling interval is 15 minutes.

Note
You can find the Default object on the Architect page in the **Other** section.

The following is a Default object for cron style pipelines:

```
{
  "name": "Default",
  "resourceRole": "DataPipelineDefaultResourceRole",
 "role": "DataPipelineDefaultRole",
  "scheduleType": "cron"
}
```

The following is a Default object for time series style pipelines:

```
{
  "name": "Default",
  "resourceRole": "DataPipelineDefaultResourceRole",
  "role": "DataPipelineDefaultRole",
  "scheduleType": "timeseries"
}
```

Resources Ignore Schedule Type

AWS Data Pipeline creates activity and data node instances at the beginning or end of the schedule interval depending on the pipeline's schedule type setting (time series style scheduling or cron style scheduling). However, AWS Data Pipeline creates `Resource` instances, such as `EC2Resource` and `EmrCluster`, at the beginning of the interval regardless of the pipeline schedule type and sets them to the WAITING_ON_DEPENDENCIES status. The actual underlying resources are not instantiated until an associated activity is scheduled.

Backfill Tasks

When you define a pipeline with a scheduled start time for the past, AWS Data Pipeline backfills the tasks in the pipeline. In that situation, AWS Data Pipeline immediately runs many instances of the tasks in the pipeline to catch up to the number of times those tasks would have run between the scheduled start time and the current time. When this happens, you see pipeline component instances running back-to-back at a greater frequency than the period value that you specified when you created the pipeline. AWS Data Pipeline returns your pipeline to the defined period only when it catches up to the number of past runs.

To minimize backfills in your development and testing phases, use a relatively short interval for `startDateTime`..`endDateTime`.

AWS Data Pipeline attempts to prevent accidental backfills by blocking pipeline activation if the pipeline component scheduledStartTime is earlier than 1 day ago.

To get your pipeline to launch immediately, set **Start Date Time** to a date one day in the past. AWS Data Pipeline starts launching the "past due" runs immediately in an attempt to address what it perceives as a backlog of work. This backfilling means you don't have to wait an hour to see AWS Data Pipeline launch its first cluster.

Maximum Resource Efficiency Using Schedules

AWS Data Pipeline allows you to maximize the efficiency of resources by supporting different schedule periods for a resource and an associated activity.

For example, consider an activity with a 20-minute schedule period. If the activity's resource were also configured for a 20-minute schedule period, AWS Data Pipeline would create three instances of the resource in an hour and consume triple the resources necessary for the task.

Instead, AWS Data Pipeline lets you configure the resource with a different schedule; for example, a one-hour schedule. When paired with an activity on a 20-minute schedule, AWS Data Pipeline creates only one resource to service all three instances of the activity in an hour, thus maximizing usage of the resource.

Protecting Against Overwriting Data

Consider a recurring import job using AWS Data Pipeline that runs multiple times per day and routes the output to the same Amazon S3 location for each run. You could accidentally overwrite your output data, unless you use a date-based expression. A date-based expression such as `s3://myBucket/#{@scheduledStartTime}` for your `S3Output.DirectoryPath` can specify a separate directory path for each period. For more information, see Schedule (p. 252).

Creating a Pipeline

AWS Data Pipeline provides several ways for you to create pipelines:

- Use the console with a template provided for your convenience. For more information, see Creating Pipelines Using Console Templates (p. 18).
- Use the console to manually add individual pipeline objects. For more information, see Creating Pipelines Using the Console Manually (p. 30).
- Use the AWS Command Line Interface (CLI) with a pipeline definition file in JSON format.
- Use an AWS SDK with a language-specific API. For more information, see Working with the API (p. 56).

Creating Pipelines Using Console Templates

The AWS Data Pipeline console provides several pre-configured pipeline definitions, known as templates. You can use templates to get started with AWS Data Pipeline quickly. You can also create templates with parameterized values. This allows you to specify pipeline objects with parameters and pre-defined attributes. You can then use a tool to create values for a specific purpose within the pipeline. This allows you to reuse pipeline definitions with different values. For more information, see Creating a Pipeline Using Parameterized Templates (p. 27).

Initialize, Create, and Schedule a Pipeline

The AWS Data Pipeline console **Create Pipeline** page allows you to create and schedule a pipeline easily.

To create and schedule a pipeline

1. Open the AWS Data Pipeline console at https://console.aws.amazon.com/datapipeline/.
2. Click either **Get started now** or **Create Pipeline**.
3. Enter a pipeline name and an optional description for the pipeline.
4. Choose **Build using Architect** to interactively create and edit nodes in a pipeline definition or **Build using a template** to select a template. For more information about templates, see Choose a Template (p. 19).

   If you use choose to use a template, the console displays a form that is specific to that template under **Parameters**. Complete the form as appropriate.
5. Choose whether to run the pipeline once on activation or on a schedule.

   If you choose to run the pipeline on a schedule:

   a. For **Run every**, choose a period for the pipeline. The start and end time must define an interval that's long enough to accommodate this period.
   b. Choose a **Starting** time. If you choose **on pipeline activation**, the pipeline uses the current activation time.
   c. Choose an **Ending** time. If you choose **never**, the pipeline runs indefinitely.
6. Select an option for **IAM Roles**. If you select **Default**, Amazon DevPay assigns its own default roles. You can optionally select **Custom** to choose other roles available to your account.
7. Click either **Edit in Architect** or **Activate**.

Choose a Template

When you choose a template, the pipeline create page populates with the parameters specified in the pipeline definition, such as custom Amazon S3 directory paths, Amazon EC2 key pair names, database connection strings, and so on. You can provide this information at pipeline creation and activation. The following templates available in the console are also available for download from the Amazon S3 bucket: `s3://datapipeline-us-east-1/templates/`.

Templates

Getting Started Using ShellCommandActivity

The **Getting Started using ShellCommandActivity** template runs a shell command script to count the number of GET requests in a log file. The output is written in a time-stamped Amazon S3 location on every scheduled run of the pipeline.

The template uses the following pipeline objects:

- ShellCommandActivity
- S3InputNode
- S3OutputNode
- Ec2Resource

Run AWS CLI Command

This template runs a user-specified AWS CLI command at scheduled intervals.

Export DynamoDB Table to S3

The **Export DynamoDB table to S3** template schedules an Amazon EMR cluster to export data from a DynamoDB table to an Amazon S3 bucket. This template uses an Amazon EMR cluster, which will be sized proportional to the value of the throughput available to the DynamoDB table. Although, you can increase IOPs on a table, this may incur additional costs while importing and exporting. Previously, export used a HiveActivity but now uses native MapReduce.

The template uses the following pipeline objects:

- EmrActivity (p. 156)
- EmrCluster (p. 210)
- DynamoDBDataNode (p. 131)
- S3DataNode (p. 143)

Import DynamoDB Backup Data from S3

The **Import DynamoDB backup data from S3** template schedules an Amazon EMR cluster to load a previously created DynamoDB backup in Amazon S3 to a DynamoDB table. Existing items in the DynamoDB table will be updated with those from the backup data and new items will be added to the table. This template uses an Amazon EMR cluster, which will be sized proportional to the value of the throughput available to the DynamoDB table. Although, you can increase IOPs on a table, this may incur additional costs while importing and exporting. Previously, import used a HiveActivity but now uses native MapReduce.

The template uses the following pipeline objects:

- EmrActivity (p. 156)
- EmrCluster (p. 210)
- DynamoDBDataNode (p. 131)
- S3DataNode (p. 143)
- S3PrefixNotEmpty (p. 232)

Run Job on an Elastic MapReduce Cluster

The **Run Job on an Elastic MapReduce Cluster** template launches an Amazon EMR cluster based on the parameters provided and starts running steps based on the specified schedule. Once the job completes, the EMR cluster is terminated. Optional bootstrap actions can be specified to install additional software or to change application configuration on the cluster.

The template uses the following pipeline objects:

- EmrActivity (p. 156)
- EmrCluster (p. 210)

Full Copy of RDS MySQL Table to S3

The **Full Copy of RDS MySQL Table to S3** template copies an entire Amazon RDS MySQL table and stores the output in an Amazon S3 location. The output is stored as a CSV file in a timestamped subfolder under the specified Amazon S3 location.

The template uses the following pipeline objects:

- CopyActivity (p. 152)
- Ec2Resource (p. 204)
- SqlDataNode (p. 147)
- S3DataNode (p. 143)

Incremental Copy of RDS MySQL Table to S3

The **Incremental Copy of RDS MySQL Table to S3** template does an incremental copy of the data from an Amazon RDS MySQL table and stores the output in an Amazon S3 location. The RDS MySQL table must have a Last Modified column. This template will copy changes that are made to the table between scheduled intervals starting from the scheduled start time. The schedule type is time series (p. 17) so if a copy was scheduled for a certain hour, Data Pipeline will copy the table rows which have a Last Modified timestamp that falls within the hour. Physical deletes to the table will not be copied. The output will be written in a timestamped subfolder under the Amazon S3 location on every scheduled run.

The template uses the following pipeline objects:

- CopyActivity (p. 152)
- Ec2Resource (p. 204)
- SqlDataNode (p. 147)
- S3DataNode (p. 143)

Load S3 Data into RDS MySQL Table

The **Load S3 Data into RDS MySQL Table** template schedules an Amazon EC2 instance to copy the CSV file from the Amazon Amazon S3 file path specified below to an Amazon RDS MYSQL table. The CSV file should not have a header row. The template updates existing entries in the RDS MySQL table with those in the Amazon S3 data and adds new entries from the Amazon S3 data to the RDS MySQL table. You can load the data into an existing table or provide an SQL query to create a new table.

The template uses the following pipeline objects:

- CopyActivity (p. 152)
- Ec2Resource (p. 204)
- SqlDataNode (p. 147)
- S3DataNode (p. 143)

Amazon RDS to Redshift Templates

The following two templates copy tables from RDS MySQL to Redshift using a translation script, which creates a Redshift table using the source table schema with the following caveats:

- If a distribution key is not specified, the first primary key from the RDS table is set as the distribution key.
- You cannot skip a column that is present in RDS MySQL table when you are doing a copy to Redshift.
- You can optional providing a RDS MySQL to Redshift column data type mapping as one of the parameters in the template. If this is specified the script will use this to create the Redshift table.

If the Overwrite_Existing Redshift insert mode is being used:

- If a distribution key is not provided, a primary key on the RDS MySQL table will be used.
- If there are composite primary keys on the table, the first one will be used as the distribution key if the distribution key is not provided. Only the first composite key is set as the primary key in the Redshift table.
- If a distribution key is not provided and there is no primary key on the RDS MySQL table, the copy operation will fail.

For more information about Redshift, see the following topics:

- Amazon Redshift Cluster and Amazon RDS Security Groups
- Redshift COPY
- Distribution styles and DISTKEY examples
- Sort Keys

The following table describes how the script translates the data types:

Data Type Translations Between MySQL and Redshift

MySQL Data Type	Redshift Data Type	Notes
TINYINT, TINYINT (size)	SMALLINT	MySQL: -128 to 127. The maximum number of digits may be specified in parenthesis Redshift: INT2. Signed two-byte integer
TINYINT UNSIGNED, TINYINT (size) UNSIGNED	SMALLINT	MySQL: 0 to 255 UNSIGNED. The maximum number of digits may be specified in parenthesis Redshift: INT2. Signed two-byte integer
SMALLINT, SMALLINT(size)	SMALLINT	MySQL: -32768 to 32767 normal. The maximum number of digits may be specified in parenthesis. Redshift: INT2. Signed two-byte integer
SMALLINT UNSIGNED, SMALLINT(size) UNSIGNED,	INTEGER	MySQL: 0 to 65535 UNSIGNED*. The maximum number of digits may be specified in parenthesis Redshift: INT4. Signed four-byte integer

MySQL Data Type	Redshift Data Type	Notes
MEDIUMINT, MEDIUMINT(size)	INTEGER	MySQL: -8388608 to 8388607. The maximum number of digits may be specified in parenthesis Redshift: INT4. Signed four-byte integer
MEDIUMINT UNSIGNED, MEDIUMINT(size) UNSIGNED	INTEGER	MySQL: 0 to 16777215. The maximum number of digits may be specified in parenthesis Redshift: INT4. Signed four-byte integer
INT, INT(size)	INTEGER	MySQL: -2147483648 to 2147483647 Redshift: INT4. Signed four-byte integer
INT UNSIGNED, INT(size) UNSIGNED	BIGINT	MySQL: 0 to 4294967295 Redshift: INT8. Signed eight-byte integer
BIGINT BIGINT(size)	BIGINT	Redshift: INT8. Signed eight-byte integer
BIGINT UNSIGNED BIGINT(size) UNSIGNED	VARCHAR(20*4)	MySQL: 0 to 18446744073709551615 Redshift: No native equivalent, so using char array.
FLOAT FLOAT(size,d) FLOAT(size,d) UNSIGNED	REAL	The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter. Redshift: FLOAT4
DOUBLE(size,d)	DOUBLE PRECISION	The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter. Redshift: FLOAT8

MySQL Data Type	Redshift Data Type	Notes
DECIMAL(size,d)	DECIMAL(size,d)	A DOUBLE stored as a string, allowing for a fixed decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter. Redshift: No native equivalent.
CHAR(size)	VARCHAR(size*4)	Holds a fixed length string, which can contain letters, numbers, and special characters. The fixed size is specified as the parameter in parenthesis. Can store up to 255 characters. Right padded with spaces. Redshift: CHAR data type does not support multibyte character so VARCHAR is used. The maximum number of bytes per character is 4 according to RFC3629, which limits the character table to U+10FFFF.
VARCHAR(size)	VARCHAR(size*4)	Can store up to 255 characters. VARCHAR does not support the following invalid UTF-8 code points: 0xD800 - 0xDFFF, (Byte sequences: ED A0 80 - ED BF BF), 0xFDD0 - 0xFDEF, 0xFFFE, and 0xFFFF, (Byte sequences: EF B7 90 - EF B7 AF, EF BF BE, and EF BF BF)
TINYTEXT	VARCHAR(255*4)	Holds a string with a maximum length of 255 characters
TEXT	VARCHAR(max)	Holds a string with a maximum length of 65,535 characters.
MEDIUMTEXT	VARCHAR(max)	0 to 16,777,215 Chars
LONGTEXT	VARCHAR(max)	0 to 4,294,967,295 Chars
BOOLEAN BOOL TINYINT(1)	BOOLEAN	MySQL: These types are synonyms for TINYINT(1). A value of zero is considered false. Nonzero values are considered true.
BINARY[(M)]	varchar(255)	M is 0 to 255 bytes, FIXED

MySQL Data Type	Redshift Data Type	Notes
VARBINARY(M)	VARCHAR(max)	0 to 65,535 bytes
TINYBLOB	VARCHAR(255)	0 to 255 bytes
BLOB	VARCHAR(max)	0 to 65,535 bytes
MEDIUMBLOB	VARCHAR(max)	0 to 16,777,215 bytes
LONGBLOB	VARCHAR(max)	0 to 4,294,967,295 bytes
ENUM	VARCHAR(255*2)	The limit is not on the length of the literal enum string, but rather on the table definition for number of enum values.
SET	VARCHAR(255*2)	Like enum.
DATE	DATE	(YYYY-MM-DD) "1000-01-01" to "9999-12-31"
TIME	VARCHAR(10*4)	(hh:mm:ss) "-838:59:59" to "838:59:59"
DATETIME	TIMESTAMP	(YYYY-MM-DD hh:mm:ss) 1000-01-01 00:00:00" to "9999-12-31 23:59:59"
TIMESTAMP	TIMESTAMP	(YYYYMMDDhhmmss) 19700101000000 to 2037+
YEAR	VARCHAR(4*4)	(YYYY) 1900 to 2155
column SERIAL	ID generation / This attribute is not needed for an OLAP data warehouse since this column is copied. SERIAL keyword is not added while translating.	SERIAL is in fact an entity named SEQUENCE. It exists independently on the rest of your table. column GENERATED BY DEFAULT equivalent to: CREATE SEQUENCE name; CREATE TABLE table (column INTEGER NOT NULL DEFAULT nextval(name));

MySQL Data Type	Redshift Data Type	Notes
column BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE	ID generation / This attribute is not needed for OLAP data warehouse since this column is copied. So SERIAL keyword is not added while translating.	SERIAL is in fact an entity named SEQUENCE. It exists independently on the rest of your table. column GENERATED BY DEFAULT equivalent to: CREATE SEQUENCE name; CREATE TABLE table (column INTEGER NOT NULL DEFAULT nextval(name));
ZEROFILL	ZEROFILL keyword is not added while translating.	INT UNSIGNED ZEROFILL NOT NULL ZEROFILL pads the displayed value of the field with zeros up to the display width specified in the column definition. Values longer than the display width are not truncated. Note that usage of ZEROFILL also implies UNSIGNED.

Full copy of RDS MySQL table to Redshift

The **Full copy of RDS MySQL table to Redshift** template copies the entire Amazon RDS MySQL table to a Redshift table by staging data in an Amazon S3 folder. The Amazon S3 staging folder must be in the same region as the Redshift cluster. A Redshift table will be created with the same schema as the source RDS MySQL table if it does not already exist. Please provide any RDS MySQL to Redshift column data type overrides you would like to apply during Redshift table creation.

The template uses the following pipeline objects:

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

Incremental copy of RDS MySQL table to Redshift

The **Incremental copy of RDS MySQL table to Redshift** template copies data from a Amazon RDS MySQL table to a Redshift table by staging data in an Amazon S3 folder. The Amazon S3 staging folder must be in the same region as the Redshift cluster. Data Pipeline uses a translation script to create a Redshift table with the same schema as the source RDS MySQL table if it does not already exist. You must provide any RDS MySQL to Redshift column data type overrides you would like to apply during Redshift table creation. This template will copy changes that are made to the RDS MySQL table between

scheduled intervals starting from the scheduled start time. Physical deletes to the RDS MySQL table will not be copied. You must provide the column name that stores the last modified time value.

The template uses the following pipeline objects:

- RDSToS3CopyActivity
- CopyActivity
- RedshiftCopyActivity
- S3DataNode
- SqlDataNode
- RedshiftDataNode
- RedshiftDatabase

Load Data from S3 Into Redshift

The **Load data from S3 into Redshift** template copies data from an Amazon S3 folder into a Redshift table. You can load the data into an existing table or provide a SQL query to create the table. The data is copied based on the Redshift COPY options provided below. The Redshift table must have the same schema as the data in Amazon S3.

The template uses the following pipeline objects:

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

Creating a Pipeline Using Parameterized Templates

You can use a parameterized template to customize a pipeline definition. This enables you to create a common pipeline definition but provide different parameters when you add the pipeline definition to a new pipeline.

Contents

Add myVariables to the Pipeline Definition

When you create the pipeline definition file, specify variables using the following syntax: #{my*Variable*}. It is required that the variable is prefixed by `my`. For example, the following pipeline definition file, `pipeline-definition.json`, includes the following variables: *`myShellCmd`*, *`myS3InputLoc`*, and *`myS3OutputLoc`*.

Note
A pipeline definition has an upper limit of 50 parameters.

```
{
```

```
  "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"
    }
  ]
}
```

Define Parameter Objects

You can create a separate file with parameter objects that defines the variables in your pipeline definition. For example, the following JSON file, `parameters.json`, contains parameter objects for the *myShellCmd*, *myS3InputLoc*, and *myS3OutputLoc* variables from the example pipeline definition above.

```
{
  "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
You could add these objects directly to the pipeline definition file instead of using a separate file.

The following table describes the attributes for parameter objects.

Parameter Attributes

Attribute	Type	Description
`id`	String	The unique identifier of the parameter. To mask the value while it is typed or displayed, add an asterisk ('*') as a prefix. For example, `*myVariable`—. Notes that this also encrypts the value before it is stored by AWS Data Pipeline.
description	String	A description of the parameter.
type	String, Integer, Double, or AWS::S3::ObjectKey	The parameter type that defines the allowed range of input values and validation rules. The default is String.
optional	Boolean	Indicates whether the parameter is optional or required. The default is `false`.
allowedValues	List of Strings	Enumerates all permitted values for the parameter.
default	String	The default value for the parameter. If you specify a value for this parameter using parameter values, it overrides the default value.

Attribute	Type	Description
isArray	Boolean	Indicates whether the parameter is an array.

Define Parameter Values

You can create a separate file to define your variables using parameter values. For example, the following JSON file, `file://values.json`, contains the value for *`myS3OutputLoc`* variable from the example pipeline definition above.

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

Submitting the Pipeline Definition

When you submit your pipeline definition, you can specify parameters, parameter objects, and parameter values. For example, you can use the put-pipeline-definition AWS CLI command as follows:

```
$ 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
A pipeline definition has an upper limit of 50 parameters. The size of the file for `parameter-values-uri` has an upper limit of 15kB.

Creating Pipelines Using the Console Manually

You can create a pipeline using the AWS Data Pipeline console without the assistance of templates. The example pipeline uses AWS Data Pipeline to copy a CSV from one Amazon S3 bucket to another on a schedule.

Prerequisites

An Amazon S3 bucket for the file copy source and destination used in the procedure. For more information, see Create a Bucket in the *Amazon Simple Storage Service Getting Started Guide*.

Tasks

- Create the Pipeline Definition (p. 30)
- Define Activities (p. 31)
- Configure the Schedule (p. 32)
- Configure Data Nodes (p. 32)
- Configure Resources (p. 33)
- Validate and Save the Pipeline (p. 33)
- Activate the Pipeline (p. 33)

Create the Pipeline Definition

Complete the initial pipeline creation screen to create the pipeline definition.

To create your pipeline definition

1. Open the AWS Data Pipeline console at https://console.aws.amazon.com/datapipeline/.
2. Click **Get started now** (if this is your first pipeline) or **Create new pipeline**.
3. In **Name**, enter a name for the pipeline (for example, `CopyMyS3Data`).
4. In **Description**, enter a description.
5. Choose a pipeline definition **Source**. You can use a template, import an existing JSON-based pipeline definition from your local file system or an Amazon S3 bucket, or create a pipeline interactively on the Architect page.

   Templates provide common scenarios encountered in Data Pipeline. You can customize a template to fit your needs by filling out the associated parameter values.

   Note
   If your existing pipeline definition contains more than one schedule, the schedule will not be visible in the create pipeline page but you can continue to the Architect page to view your schedules.
6. In **Pipeline Configuration**, if you choose to enable logging, select a bucket in Amazon S3 to store logs for this pipeline.
7. Leave the **Schedule** fields set to their default values.
8. Leave **IAM roles** set to **Default**.

   Alternatively, if you created your own IAM roles and would like to use them, click **Custom** and select them from the **Pipeline role** and **EC2 instance role** lists.
9. Click **Create**.

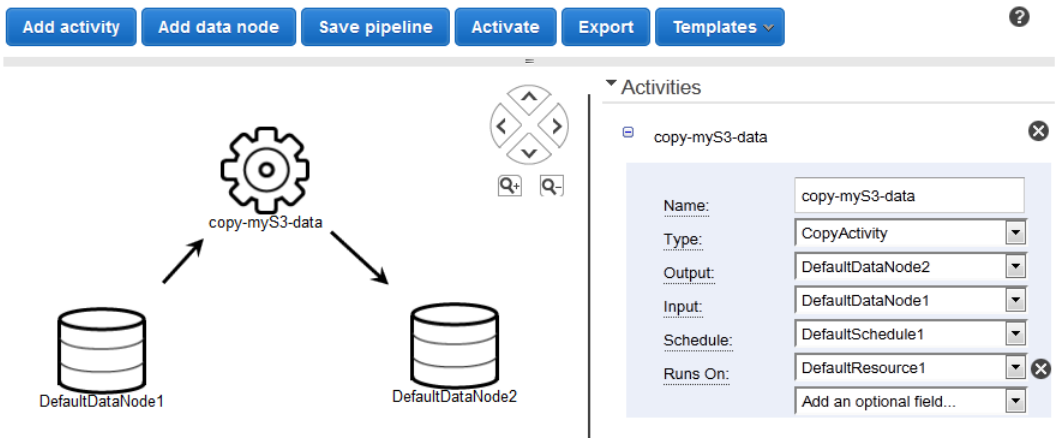
Define Activities

Add `Activity` objects to your pipeline definition. When you define an `Activity` object, you must also define the objects that AWS Data Pipeline needs to perform this activity.

To define activities for your pipeline

1. On the pipeline page, click **Add activity**.
2. From the **Activities** pane, in **Name**, enter a name for the activity (for example, `copy-myS3-data`).
3. In **Type**, select **CopyActivity**.
4. In **Schedule**, select **Create new: Schedule**.
5. In **Input**, select **Create new: DataNode**.
6. In **Output**, select **Create new: DataNode**.
7. In **Add an optional field**, select **RunsOn**.
8. In **Runs On**, select **Create new: Resource**.
9. In the left pane, separate the icons by dragging them apart.

   This is a graphical representation of the pipeline. The arrows indicate the connection between the various objects. Your pipeline should look similar to the following image.



Configure the Schedule

Configure the run date and time for your pipeline. Note that AWS Data Pipeline supports the date and time expressed in "YYYY-MM-DDTHH:MM:SS" format in UTC/GMT only.

To configure the run date and time for your pipeline

1. On the pipeline page, in the right pane, expand the **Schedules** pane.
2. Enter a schedule name for this activity (for example, `copy-myS3-data-schedule`).
3. In **Start Date Time**, select the date from the calendar, and then enter the time to start the activity.
4. In **Period**, enter the duration for the activity (for example, `1`), and then select the period category (for example, `Days`).
5. (Optional) To specify the date and time to end the activity, in **Add an optional field**, select **End Date Time**, and enter the date and time.

   To get your pipeline to launch immediately, set **Start Date Time** to a date one day in the past. AWS Data Pipeline then starts launching the "past due" runs immediately in an attempt to address what it perceives as a backlog of work. This backfilling means you don't have to wait an hour to see AWS Data Pipeline launch its first cluster.

Configure Data Nodes

Configure the input and the output data nodes for your pipeline.

To configure the input and output data nodes of your pipeline

1. On the pipeline page, in the right pane, click **DataNodes**.
2. Under `DefaultDataNode1`, in **Name**, enter a name for the Amazon S3 bucket to use as your input node (for example, `MyS3Input`).
3. In **Type**, select **S3DataNode**.
4. In **Schedule**, select **copy-myS3-data-schedule**.
5. In **Add an optional field**, select **File Path**.
6. In **File Path**, enter the path to your Amazon S3 bucket (for example, `s3://my-data-pipeline-input/data`).
7. Under `DefaultDataNode2`, in **Name**, enter a name for the Amazon S3 bucket to use as your output node (for example, `MyS3Output`).

8. In **Type**, select **S3DataNode**.
9. In **Schedule**, select **copy-myS3-data-schedule**.
10. In **Add an optional field**, select **File Path**.
11. In **File Path**, enter the path to your Amazon S3 bucket (for example, `s3://my-data-pipeline-output/data`).

Configure Resources

Configure the resource that AWS Data Pipeline must use to perform the copy activity, an EC2 instance.

To configure an EC2 instance for your pipeline

1. On the pipeline page, in the right pane, click **Resources**.
2. In **Name**, enter a name for your resource (for example, `CopyDataInstance`).
3. In **Type**, select **Ec2Resource**.
4. [EC2-VPC] In **Add an optional field**, select **Subnet Id**.
5. [EC2-VPC] In **Subnet Id**, enter the ID of the subnet.
6. In **Schedule**, select **copy-myS3-data-schedule**.
7. Leave **Role** and **Resource Role** set to their default values.

   Alternatively, if you created your own IAM roles and would like to use them, click **Custom** and select them from the **Pipeline role** and **EC2 instance role** lists.

Validate and Save the Pipeline

You can save your pipeline definition at any point during the creation process. As soon as you save your pipeline definition, AWS Data Pipeline looks for syntax errors and missing values in your pipeline definition. If your pipeline is incomplete or incorrect, AWS Data Pipeline generates validation errors and warnings. Warning messages are informational only, but you must fix any error messages before you can activate your pipeline.

To save and validate your pipeline

1. Choose **Save pipeline**.
2. AWS Data Pipeline validates your pipeline definition and returns either success or error or warning messages. If you get an error message, choose **Close** and then, in the right pane, choose **Errors/Warnings**.
3. The **Errors/Warnings** pane lists the objects that failed validation. Choose the plus (**+**) sign next to the object names and look for an error message in red.
4. When you see an error message, go to the specific object pane where you see the error and fix it. For example, if you see an error message in the **DataNodes** object, go to the **DataNodes** pane to fix the error.
5. After you fix the errors listed in the **Errors/Warnings** pane, choose **Save Pipeline**.
6. Repeat the process until your pipeline validates successfully.

Activate the Pipeline

Activate your pipeline to start creating and processing runs. The pipeline starts based on the schedule and period in your pipeline definition.

Important
If activation succeeds, your pipeline is running and might incur usage charges. For more information, see AWS Data Pipeline pricing. To stop incurring usage charges for AWS Data Pipeline, delete your pipeline.

To activate your pipeline

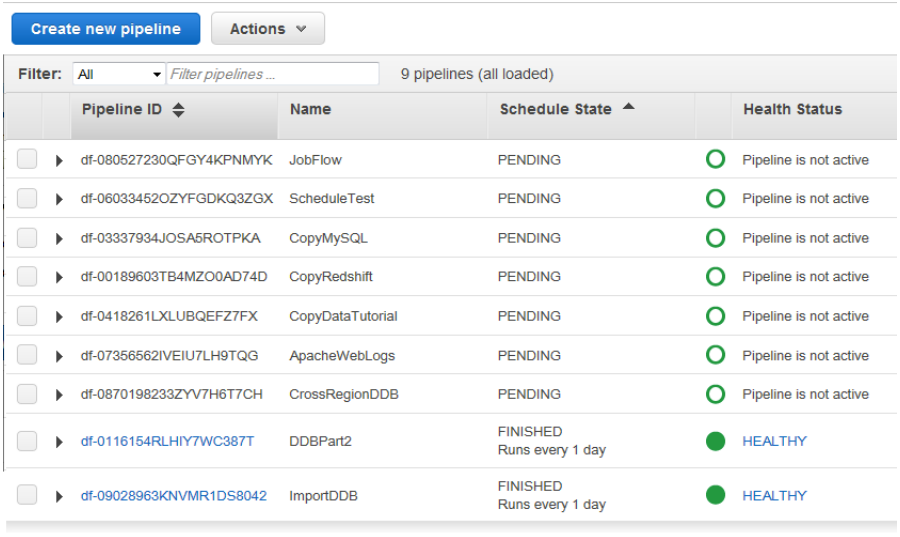
1. Choose **Activate**.
2. In the confirmation dialog box, choose **Close**.

Viewing Your Pipelines

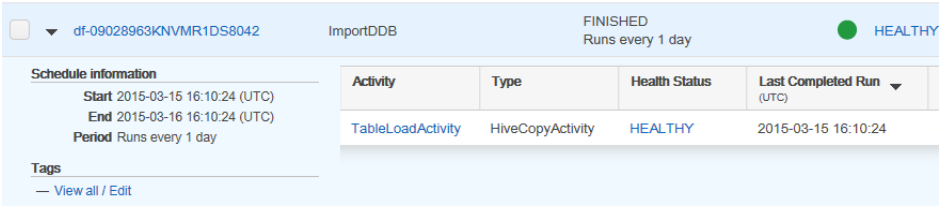
You can view your pipelines using the console or the command line interface (CLI).

To view your pipelines using the console

Open the AWS Data Pipeline console. If you have created any pipelines in that region, the console displays them in a list. Otherwise, you see a welcome screen.



To view information about a pipeline, click the arrow. The console displays information about the schedule, activities, and tags for the pipeline. For more information about the health status, see Interpreting Pipeline and Component Health State (p. 36).



To view your pipelines using the AWS CLI

Use the following list-pipelines command to list your pipelines:

```
aws datapipeline list-pipelines
```

Interpreting Pipeline Status Codes

The status levels displayed in the AWS Data Pipeline console and CLI indicate the condition of a pipeline and its components. The pipeline status is simply an overview of a pipeline; to see more information, view the status of individual pipeline components.

A pipeline has a `SCHEDULED` status if it is ready (the pipeline definition passed validation), currently performing work, or finished performing work. A pipeline has a `PENDING` status if it is not activated or not able to perform work (for example, the pipeline definition failed validation.)

A pipeline is considered inactive if its status is `PENDING`, `INACTIVE`, or `FINISHED`. Inactive pipelines incur a charge (for more information, see Pricing).

Status Codes

`ACTIVATING`

The component or resource is being started, such as an EC2 instance.

`CANCELED`

The component was canceled by a user or AWS Data Pipeline before it could run. This can happen automatically when a failure occurs in a different component or resource that this component depends on.

`CASCADE_FAILED`

The component or resource was canceled as a result of a cascade failure from one of its dependencies, but the component was probably not the original source of the failure.

`DEACTIVATING`

The pipeline is being deactivated.

`FAILED`

The component or resource encountered an error and stopped working. When a component or resource fails, it can cause cancelations and failures to cascade to other components that depend on it.

`FINISHED`

The component completed its assigned work.

`INACTIVE`

The pipeline was deactivated.

`PAUSED`

The component was paused and is not currently performing its work.

`PENDING`

The pipeline is ready to be activated for the first time.

`RUNNING`

The resource is running and ready to receive work.

`SHUTTING_DOWN`

The resource is shutting down after successfully completing its work.

`SKIPPED`

The component skipped intervals of execution after the pipeline was activated using a timestamp that is later than the current schedule.

`TIMEDOUT`

The resource exceeded the `terminateAfter` threshold and was stopped by AWS Data Pipeline. After the resource reaches this status, AWS Data Pipeline ignores the `actionOnResourceFailure`, `retryDelay`, and `retryTimeout` values for that resource. This status applies only to resources.

`VALIDATING`

The pipeline definition is being validated by AWS Data Pipeline.

`WAITING_FOR_RUNNER`

The component is waiting for its worker client to retrieve a work item. The component and worker client relationship is controlled by the `runsOn` or `workerGroup` fields defined by that component.

`WAITING_ON_DEPENDENCIES`

The component is verifying that its default and user-configured preconditions are met before performing its work.

Interpreting Pipeline and Component Health State

Each pipeline and component within that pipeline returns a health status of `HEALTHY`, `ERROR`, `"-"`, `No Completed Executions`, or `No Health Information Available`. A pipeline only has a health state after a pipeline component has completed its first execution or if component preconditions have failed. The health status for components aggregates into a pipeline health status in that error states are visible first when you view your pipeline execution details.

Pipeline Health States

`HEALTHY`

The aggregate health status of all components is `HEALTHY`. This means at least one component must have successfully completed. You can click on the `HEALTHY` status to see the most recent successfully-completed pipeline component instance on the **Execution Details** page.

`ERROR`

At least one component in the pipeline has a health status of `ERROR`. You can click on the `ERROR` status to see the most recent failed pipeline component instance on the **Execution Details** page.

`No Completed Executions` or `No Health Information Available`.

No health status was reported for this pipeline.

Note
While components update their health status almost immediately, it may take up to five minutes for a pipeline health status to update.

Component Health States

`HEALTHY`

A component (`Activity` or `DataNode`) has a health status of `HEALTHY` if it has completed a successful execution where it was marked with a status of `FINISHED` or `MARK_FINISHED`. You can click on the name of the component or the `HEALTHY` status to see the most recent successfully-completed pipeline component instances on the **Execution Details** page.

`ERROR`

An error occurred at the component level or one of its preconditions failed. Statuses of `FAILED`, `TIMEOUT`, or `CANCELED` trigger this error. You can click on the name of the component or the `ERROR` status to see the most recent failed pipeline component instance on the **Execution Details** page.

`No Completed Executions` or `No Health Information Available`

No health status was reported for this component.

Viewing Your Pipeline Definitions

Use the AWS Data Pipeline console or the command line interface (CLI) to view your pipeline definition. The console shows a graphical representation, while the CLI prints a pipeline definition file, in JSON format. For information about the syntax and usage of pipeline definition files, see Pipeline Definition File Syntax (p. 54).

To view a pipeline definition using the console

1. On the **List Pipelines** page, click the **Pipeline ID** for the desired pipeline, which displays the pipeline **Architect** page.
2. On the pipeline **Architect** page, click the object icons in the design pane to expand the corresponding section in the right pane.

   Alternatively, expand one of the sections in the right pane to view its objects and their associated fields.
3. If your pipeline definition graph does not fit in the design pane, use the pan buttons on the right side of the design pane to slide the canvas.

4. You can also view the entire text pipeline definition by clicking **Export**. A dialog appears with the JSON pipeline definition.

If you are using the CLI, it's a good idea to retrieve the pipeline definition before you submit modifications, because it's possible that another user or process changed the pipeline definition after you last worked with it. By downloading a copy of the current definition and using that as the basis for your modifications, you can be sure that you are working with the most recent pipeline definition. It's also a good idea to retrieve the pipeline definition again after you modify it, so that you can ensure that the update was successful.

If you are using the CLI, you can get two different versions of your pipeline. The `active` version is the pipeline that is currently running. The `latest` version is a copy that's created when you edit a running pipeline. When you upload the edited pipeline, it becomes the `active` version and the previous `active` version is no longer available.

To get a pipeline definition using the AWS CLI

To get the complete pipeline definition, use the get-pipeline-definition command. The pipeline definition is printed to standard output (stdout).

The following example gets the pipeline definition for the specified pipeline.

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

To retrieve a specific version of a pipeline, use the `--version` option. The following example retrieves the `active` version of the specified pipeline.

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

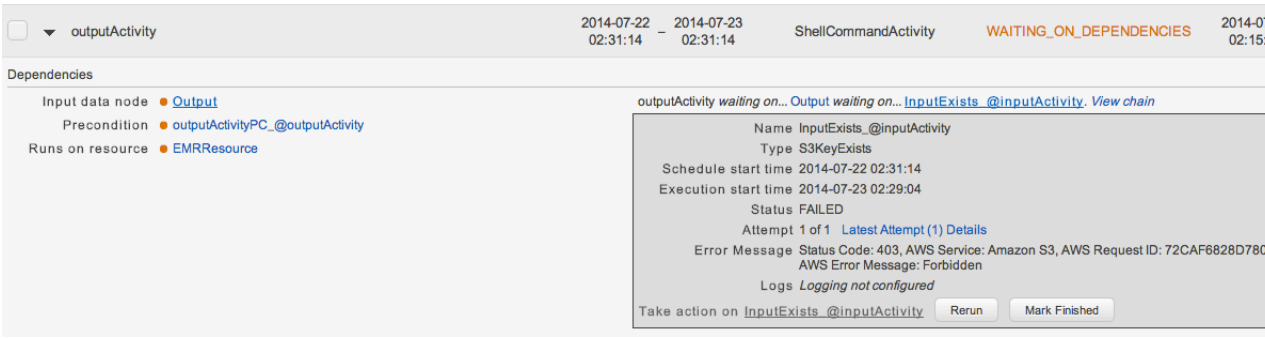
Viewing Pipeline Instance Details

You can monitor the progress of your pipeline. For more information about instance status, see Interpreting Pipeline Status Details (p. 272). For more information about troubleshooting failed or incomplete instance runs of your pipeline, see Resolving Common Problems (p. 273).

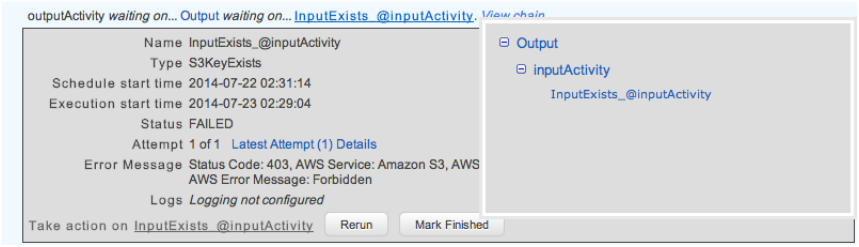
To monitor the progress of a pipeline using the console

1. On the **List Pipelines** page, in the **Pipeline ID** column, click the arrow for your pipeline and click **View execution details**.
2. The **Execution details** page lists the name, type, status, and schedule information of each component.

   You can then click on the arrow for each component name to view dependency information for that component.



   In the inline summary, you can view instance details, re-run an activity, mark it as `FINISHED`, or explore the dependency chain.



   Note
   If you do not see runs listed, check when your pipeline was scheduled. Either change **End (in UTC)** to a later date or change **Start (in UTC)** an earlier date, and then click **Update**.
3. If the **Status** column of all components in your pipeline is `FINISHED`, your pipeline has successfully completed the activity. You should receive an email about the successful completion of this task, to the account that you specified to receive Amazon SNS notifications.

   You can also check the content of your output data node.
4. If the **Status** column of any component in your pipeline is not `FINISHED`, either your pipeline is waiting for some dependency or it has failed. To troubleshoot failed or the incomplete instance runs, use the following procedure.
5. Click the triangle next to an component or activity.

   If the status of the instance is `FAILED`, the **Attempts** box has an **Error Message** indicating the reason for failure under the latest attempt. For example, `Status Code: 403, AWS Service: Amazon S3, AWS`

`Request ID: 1A3456789ABCD, AWS Error Code: null, AWS Error Message: Forbidden.` You can also click on **More...** in the **Details** column to view the instance details of this attempt.

6. To take an action on your incomplete or failed component, click an action button (**Rerun**, **Mark Finished**, or **Cancel**).

To monitor the progress of a pipeline using the AWS CLI

To retrieve pipeline instance details, such as a history of the times that a pipeline has run, use the list-runs command. This command enables you to filter the list of runs returned based on either their current status or the date-range in which they were launched. Filtering the results is useful because, depending on the pipeline's age and scheduling, the run history can be very large.

The following example retrieves information for all runs.

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

The following example retrieves information for all runs that have completed.

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

The following example retrieves information for all runs launched in the specified time frame.

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

Viewing Pipeline Logs

Pipeline-level logging is supported at pipeline creation by specifying an Amazon S3 location in either the console or with a `pipelineLogUri` in the default object in SDK/CLI. The directory structure for each pipeline within that URI is like the following:

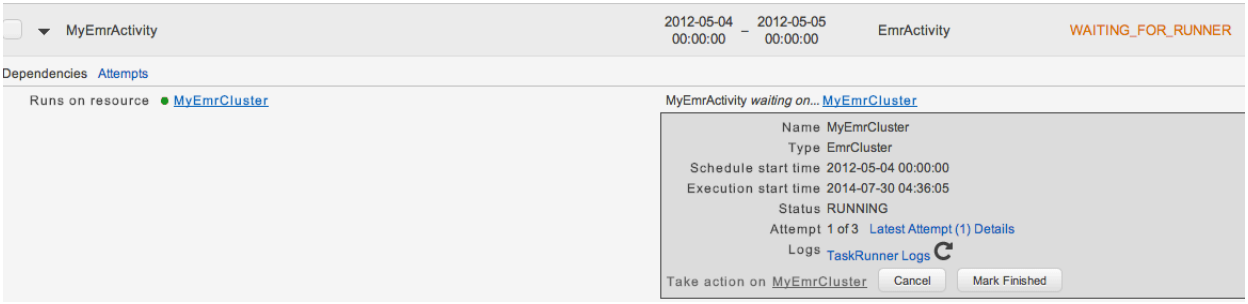
```
pipelineId
    -componentName
        -instanceId
            -attemptId
```

For pipeline, `df-00123456ABC7DEF8HIJK`, the directory structure looks like:

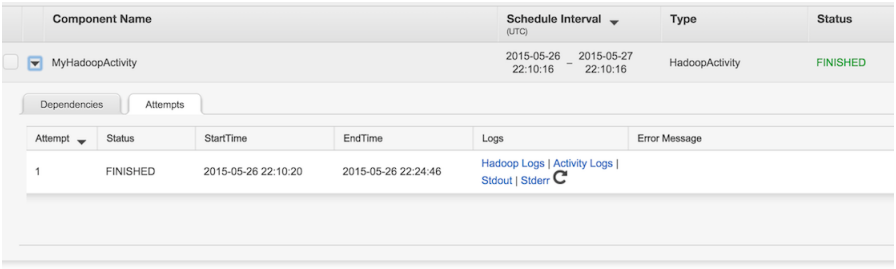
```
df-00123456ABC7DEF8HIJK
    -ActivityId_fXNzc
        -@ActivityId_fXNzc_2014-05-01T00:00:00
            -@ActivityId_fXNzc_2014-05-01T00:00:00_Attempt=1
```

For `ShellCommandActivity`, logs for `stderr` and `stdout` associated with these activities are stored in the directory for each attempt.

For resources like, `EmrCluster`, where an `emrLogUri` is set, that value takes precedence. Otherwise, resources (including TaskRunner logs for those resources) will follow the above pipeline logging structure. You may view these logs for each component in the **Execution Details** page for your pipeline by viewing a components details and clicking on the link for logs:



You can also view logs for each attempt. For example, to view logs for a `HadoopActivity`, you can click the pipeline **Attempts** tab for your activity. Hadoop Logs gives the logs created by Hadoop jobs.



Editing Your Pipeline

If you need to change some aspect of one of your pipelines, you can update its pipeline definition. After you change a pipeline that is running, you must re-activate the pipeline for your changes to take effect. In addition, you can re-run one or more pipeline components.

Contents

Limitations

Before you activate a pipeline, you can make any changes to it. After you activate a pipeline, you can edit the pipeline with the following restrictions. The changes you make apply to new runs of the pipeline objects after you save them and then activate the pipeline again.

- You can't remove an object
- You can't change the schedule period of an existing object
- You can't add, delete, or modify reference fields in an existing object
- You can't reference an existing object in an output field of a new object
- You can't change the scheduled start date of an object (instead, activate the pipeline with a specific date and time)

Editing a Pipeline Using the Console

You can edit a pipeline using the AWS Management Console.

To edit a pipeline using the console

1. On the **List Pipelines** page, check the **Pipeline ID** and **Name** columns for your pipeline, and then click your Pipeline ID.
2. To complete or modify your pipeline definition:
   a. On the pipeline (Architect) page, click the object panes in the right pane and finish defining the objects and fields of your pipeline definition. If you are modifying an active pipeline, some fields are grayed out and can't be modified. It might be easier to clone the pipeline and edit the copy, depending on the changes you need to make. For more information, see Cloning Your Pipeline (p. 42).
   b. Click **Save pipeline**. If there are validation errors, fix them and save the pipeline again.
3. After you've saved your pipeline definition with no validation errors, click **Activate**.
4. In the **List Pipelines** page, check whether your newly-created pipeline is listed and the **Schedule State** column displays `SCHEDULED`.
5. After editing an active pipeline, you might decide to rerun one or more pipeline components.

   On the **List Pipelines** page, in the detail dropdown of your pipeline, click **View execution details**.

   a. On the **Execution details** page, choose a pipeline component dropdown from the list to view the details for a component.
   b. Click **Rerun**.
   c. At the confirmation prompt, click **Continue**.

      The changed pipeline component and any dependencies will change status. For example, resources change to the `CREATING` status and activities change to the `WAITING_FOR_RUNNER` status.

Editing a Pipeline Using the AWS CLI

You can edit a pipeline using the command line tools.

First, download a copy of the current pipeline definition using the get-pipeline-definition command. By doing this, you can be sure that you are modifying the most recent pipeline definition. The following example uses prints the pipeline definition to standard output (stdout).

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

Save the pipeline definition to a file and edit it as needed. Update your pipeline definition using the put-pipeline-definition command. The following example uploads the updated pipeline definition file.

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

You can retrieve the pipeline definition again using the `get-pipeline-definition` command to ensure that the update was successful. To activate the pipeline, use the following activate-pipeline command:

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

If you prefer, you can activate the pipeline from a specific date and time, using the `--start-timestamp` option as follows:

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

To re-run one or more pipeline components, use the set-status command.

Cloning Your Pipeline

Cloning makes a copy of a pipeline and allows you to specify a name for the new pipeline. You can clone a pipeline that is in any state, even if it has errors; however, the new pipeline remains in the `PENDING` state until you manually activate it. For the new pipeline, the clone operation uses the latest version of the original pipeline definition rather than the active version. In the clone operation, the full schedule from the original pipeline is not copied into the new pipeline, only the period setting.

Note
You can't clone a pipeline using the command line interface (CLI).

To clone a pipeline using the console

1. In the **List Pipelines** page, select the pipeline to clone.
2. Click **Actions**, and then click **Clone**.
3. In the **Clone a Pipeline** dialog box, enter a name for the new pipeline and click **Clone**.
4. In the **Schedule** pane, specify a schedule for the new pipeline.
5. To activate the new pipeline, click **Actions**, and then click **Activate**.

Tagging Your Pipeline

Tags are case-sensitive key/value pairs that consist of a key and an optional value, both defined by the user. You can apply up to 10 tags to each pipeline. Tag keys must be unique for each pipeline. If you add a tag with a key that is already associated with the pipeline, it updates the value of that tag.

Applying a tag to a pipeline also propagates the tags to its underlying resources (for example, EMR clusters and EC2 instances). However, it does not apply these tags to resources in a `FINISHED` or otherwise terminated state.

When you are finished with a tag, you can remove it from your pipeline.

To tag your pipeline using the console

1. Open the AWS Data Pipeline console at https://console.aws.amazon.com/datapipeline/.
2. On the **List Pipelines** page, in the **Pipeline ID** column, click the expand arror next to your pipeline, and then click **View all/Edit** under **Tags**.
3. In the **View all / Edit** dialog box, do the following:
   a. Specify a key and a value for each tag that you'd like to add.
   b. Click the remove icon of any tags that you'd like to remove.
   c. Click **Save**.

To tag your pipeline using the AWS CLI

To add tags to a new pipeline, add the `--tags` option to your create-pipeline command. For example, the following option creates a pipeline with two tags, an `environment` tag with a value of `production`, and an `owner` tag with a value of `sales`.

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

To add tags to an existing pipeline, use the add-tags command as follows:

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

To remove tags from an existing pipeline, use the remove-tags command as follows:

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

Deactivating Your Pipeline

Deactivating a running pipeline pauses the pipeline execution. To resume pipeline execution, you can activate the pipeline. This enables you to make changes. For example, if you are writing data to a database that is scheduled to undergo maintenance, you can deactivate the pipeline, wait for the maintenance to complete, and then activate the pipeline.

When you deactivate a pipeline, you can specify what happens to running activities. By default, these activities are canceled immediately. Alternatively, you can have AWS Data Pipeline wait until the activities finish before deactivating the pipeline.

When you activate a deactivated pipeline, you can specify when it resumes. For example, using the AWS Management Console, you can resume after the last completed run, from the current time, or from a specified date and time. Using the AWS CLI or the API, the pipeline resumes from the last completed execution by default, or you can specify the date and time to resume the pipeline.

Contents

Deactivate Your Pipeline Using the Console

Use the following procedure to deactivate a running pipeline.

To deactivate a pipeline

1. In the **List Pipelines** page, select the pipeline to deactivate.
2. Click **Actions**, and then click **Deactivate**.
3. In the **Deactivate a Pipeline** dialog box, select an option, and then click **Deactivate**.
4. When prompted for confirmation, click **Deactivate**.

When you are ready to resume the pipeline runs, use the following procedure to activate the deactivated pipeline.

To activate a pipeline

1. In the **List Pipelines** page, select the pipeline to activate.

2. Click **Actions**, and then click **Activate**.
3. In the **Activate a Pipeline** dialog box, select an option, and then choose **Activate**.

Deactivate Your Pipeline Using the AWS CLI

Use the following deactivate-pipeline command to deactivate a pipeline:

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

To deactivate the pipeline only after all running activities finish, add the `--no-cancel-active` option, as follows:

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

When you are ready, you can resume the pipeline execution where it left off using the following activate-pipeline command:

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

To start the pipeline from a specific date and time, add the `--start-timestamp` option, as follows:

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

Deleting Your Pipeline

When you no longer require a pipeline, such as a pipeline created during application testing, you should delete it to remove it from active use. Deleting a pipeline puts it into a deleting state. When the pipeline is in the deleted state, its pipeline definition and run history are gone. Therefore, you can no longer perform operations on the pipeline, including describing it.

> **Important**
> You can't restore a pipeline after you delete it, so be sure that you won't need the pipeline in the future before you delete it.

To delete a pipeline using the console

1. In the **List Pipelines** page, select the pipeline.
2. Click **Actions**, and then click **Delete**.
3. When prompted for confirmation, click **Delete**.

To delete a pipeline using the AWS CLI

To delete a pipeline, use the delete-pipeline command. The following command deletes the specified pipeline.

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

Staging Data and Tables with Pipeline Activities

AWS Data Pipeline can stage input and output data in your pipelines to make it easier to use certain activities, such as `ShellCommandActivity` and `HiveActivity`. Data staging is when AWS Data Pipeline copies data from the input data node to the resource executing the activity, and similarly from the resource to the output data node. The staged data on the Amazon EMR or Amazon EC2 resource is available by using special variables in the activity's shell commands or hive scripts. Table staging is similar to data staging, except the staged data takes the form of database tables, specifically. AWS Data Pipeline supports the following staging scenarios:

- Data staging with `ShellCommandActivity`
- Table staging with Hive and staging-supported data nodes
- Table staging with Hive and staging-unsupported data nodes

Note
Staging only functions when the `stage` field is set to `true` on an activity, such as ShellCommandActivity. For more information, see ShellCommandActivity (p. 194).

In addition, data nodes and activities can relate in four ways:

Staging data locally on a resource

The input data automatically copies into the resource local file system. Output data automatically copies from the resource local file system to the output data node. For example, when you configure `ShellCommandActivity` inputs and outputs with staging = true, the input data is available as INPUT*x*_STAGING_DIR and output data is available as OUTPUT*x*_STAGING_DIR, where *x* is the number of input or output.

Staging input and output definitions for an activity

The input data format (column names and table names) automatically copies into the activity's resource. For example, when you configure `HiveActivity` with staging = true. The data format specified on the input `S3DataNode` is used to stage the table definition from the Hive table.

Staging not enabled

The input and output objects and their fields are available for the activity, but the data itself is not. For example, `EmrActivity` by default or when you configure other activities with staging = false. In this configuration, the data fields are available for the activity to make a reference to them using the AWS Data Pipeline expression syntax, and this only occurs when the dependency is satisfied. This serves as dependency checking only. Code in the activity is responsible for copying the data from the input to the resource running the activity

Dependency relationship between objects

There is a depends-on relationship between two objects, which results in a similar situation to when staging is not enabled. This causes a data node or activity to act as a precondition for the execution of another activity.

Data Staging with ShellCommandActivity

Consider a scenario using a `ShellCommandActivity` with `S3DataNode` objects as data input and output. AWS Data Pipeline automatically stages the data nodes to make them accessible to the shell command as if they were local file folders using the environment variables `${INPUT1_STAGING_DIR}` and `${OUTPUT1_STAGING_DIR}` as shown in the following example. The numeric portion of the variables named `INPUT1_STAGING_DIR` and `OUTPUT1_STAGING_DIR` increment depending on the number of data nodes your activity references.

Note
This scenario only works as described if your data inputs and outputs are `S3DataNode` objects. Additionally, output data staging is allowed only when `directoryPath` is set on the output `S3DataNode` object.

```
{
  "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')}"
  }
},
...
```

Table Staging with Hive and Staging-supported Data Nodes

Consider a scenario using a `HiveActivity` with `S3DataNode` objects as data input and output. AWS Data Pipeline automatically stages the data nodes to make them accessible to the Hive script as if they were Hive tables using the variables `${input1}` and `${output1}` as shown in the following example `HiveActivity`. The numeric portion of the variables named `input` and `output` increment depending on the number of data nodes your activity references.

Note
This scenario only works as described if your data inputs and outputs are `S3DataNode` or `MySqlDataNode` objects. Table staging is not supported for `DynamoDBDataNode`.

```
{
  "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"
  }
},
...
```

Table Staging with Hive and Staging-unsupported Data Nodes

Consider a scenario using a `HiveActivity` with `DynamoDBDataNode` as data input and an `S3DataNode` object as the output. No data staging is available for `DynamoDBDataNode`, therefore you must first manually create the table within your hive script, using the variable name `#{input.tableName}` to refer to the DynamoDB table. Similar nomenclature applies if the DynamoDB table is the output, except you use variable `#{output.tableName}`. Staging is available for the output `S3DataNode` object in this example, therefore you can refer to the output data node as `${output1}`.

Note
In this example, the table name variable has the # (hash) character prefix because AWS Data Pipeline uses expressions to access the `tableName` or `directoryPath`. For more information about how expression evaluation works in AWS Data Pipeline, see Expression Evaluation (p. 123).

```
{
  "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"
  }
},
...
```

Launching Resources for Your Pipeline into a VPC

Pipelines can launch Amazon EC2 instances and Amazon EMR clusters into a virtual private cloud (VPC). First, create a VPC and subnets using Amazon VPC and configure the VPC so that instances in the VPC can access Amazon S3. Next, set up a security group that grants Task Runner access to your data sources. Finally, specify a subnet from the VPC when you configure your instances and clusters and when you create your data sources.

Note that if you have a default VPC in a region, it's already configured to access other AWS services. When you launch a resource, we'll automatically launch it into your default VPC.

For more information about VPCs, see the Amazon VPC User Guide.

Contents

Create and Configure a VPC

A VPC that you create must have a subnet, an Internet gateway, and a route table for the subnet with a route to the Internet gateway so that instances in the VPC can access Amazon S3. (If you have a default VPC, it is already configured this way.) The easiest way to create and configure your VPC is to use the VPC wizard, as shown in the following procedure.

To create and configure your VPC using the VPC wizard

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. From the navigation bar, use the region selector to select the region for your VPC. You'll launch all instances and clusters into this VPC, so select the region that makes sense for your pipeline.

3. Click **VPC Dashboard** in the navigation pane.
4. Locate the **Your Virtual Private Cloud** area of the dashboard and click **Get started creating a VPC**, if you have no VPC resources, or click **Start VPC Wizard**.
5. Select the first option, **VPC with a Single Public Subnet Only**, and then click **Continue**.
6. The confirmation page shows the CIDR ranges and settings that you've chosen. Verify that **Enable DNS hostnames** is **Yes**. Make any other changes that you need, and then click **Create VPC** to create your VPC, subnet, Internet gateway, and route table.
7. After the VPC is created, click **Your VPCs** in the navigation pane and select your VPC from the list.
   - On the **Summary** tab, make sure that both **DNS resolution** and **DNS hostnames** are **yes**.
   - Click the identifier for the DHCP options set. Make sure that **domain-name-servers** is `AmazonProvidedDNS` and **domain-name** is `ec2.internal` for the US East (N. Virginia) region and *`region-name`*`.compute.internal` for all other regions. Otherwise, create a new options set with these settings and associate it with the VPC. For more information, see Working with DHCP Options Sets in the *Amazon VPC User Guide*.

If you prefer to create the VPC, subnet, Internet gateway, and route table manually, see Creating a VPC and Adding an Internet Gateway to Your VPC in the *Amazon VPC User Guide*.

Set Up Connectivity Between Resources

Security groups act as a virtual firewall for your instances to control inbound and outbound traffic. You must grant Task Runner access to your data sources.

For more information about security groups, see Security Groups for Your VPC in the *Amazon VPC User Guide*.

First, identify the security group or IP address used by the resource running Task Runner.

- If your resource is of type EmrCluster (p. 210), Task Runner runs on the cluster by default. We create security groups named `ElasticMapReduce-master` and `ElasticMapReduce-slave` when you launch the cluster. You'll need the IDs of these security groups later on.

  To get the IDs of the security groups for a cluster in a VPC

  1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
  2. In the navigation pane, click **Security Groups**.
  3. If you have a lengthy list of security groups, you can click the **Name** column to sort your security groups by name. (If you don't see a **Name** column, click the **Show/Hide Columns** icon, and then click **Name**.)
  4. Note the IDs of the `ElasticMapReduce-master` and `ElasticMapReduce-slave` security groups.
- If your resource is of type Ec2Resource (p. 204), Task Runner runs on the EC2 instance by default. Create a security group for the VPC and specify it when you launch the EC2 instance. You'll need the ID of this security group later on.

  To create a security group for an EC2 instance in a VPC

  1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
  2. In the navigation pane, click **Security Groups**.
  3. Click **Create Security Group**.
  4. Specify a name and description for the security group.
  5. Select your VPC from the list, and then click **Create**.
  6. Note the ID of the new security group.

- If you are running Task Runner on your own computer, note its public IP address, in CIDR notation. If the computer is behind a firewall, note the entire address range of its network. You'll need this address later on.

Next, create rules in the resource security groups that allow inbound traffic for the data sources Task Runner must access. For example, if Task Runner must access a Amazon Redshift cluster, the security group for the Amazon Redshift cluster must allow inbound traffic from the resource.

To add a rule to the security group for an RDS database

1. Open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, click **Instances**.
3. Click the details icon for the DB instance. Under **Security and Network**, click the link to the security group, which takes you to the Amazon EC2 console. If you're using the old console design for security groups, switch to the new console design by clicking the icon that's displayed at the top of the console page.
4. From the **Inbound** tab, click **Edit** and then click **Add Rule**. Specify the database port that you used when you launched the DB instance. Start typing the ID of the security group or IP address used by the resource running Task Runner in **Source**.
5. Click **Save**.

To add a rule to the security group for a Amazon Redshift cluster

1. Open the Amazon Redshift console at https://console.aws.amazon.com/redshift/.
2. In the navigation pane, click **Clusters**.
3. Click the details icon for the cluster. Under **Cluster Properties**, note the name or ID of the security group, and then click **View VPC Security Groups**, which takes you to the Amazon EC2 console. If you're using the old console design for security groups, switch to the new console design by clicking the icon that's displayed at the top of the console page.
4. Select the security group for the cluster.
5. From the **Inbound** tab, click **Edit** and then click **Add Rule**. Specify the type, protocol, and port range. Start typing the ID of the security group or IP address used by the resource running Task Runner in **Source**.
6. Click **Save**.

Configure the Resource

To launch a resource into a subnet of a nondefault VPC or a nondefault subnet of a default VPC, you must specify the subnet using the `subnetId` field when you configure the resource. If you have a default VPC and you don't specify `subnetId`, we'll launch the resource into the default subnet of the default VPC.

Example EmrCluster

The following example object launches an Amazon EMR cluster into a nondefault VPC.

```
{
  "id" : "MyEmrCluster",
  "type" : "EmrCluster",
  "keyPair" : "my-key-pair",
  "masterInstanceType" : "m1.xlarge",
  "coreInstanceType" : "m1.small",
  "coreInstanceCount" : "10",
  "taskInstanceType" : "m1.small",
```

```
  "taskInstanceCount": "10",
  "subnetId": "subnet-12345678"
}
```

For more information, see EmrCluster (p. 210).

Example Ec2Resource

The following example object launches an EC2 instance into a nondefault VPC. Notice that you must specify security groups for an instance in a nondefault VPC using their IDs, not their names.

```
{
  "id" : "MyEC2Resource",
  "type" : "Ec2Resource",
  "actionOnTaskFailure" : "terminate",
  "actionOnResourceFailure" : "retryAll",
  "maximumRetries" : "1",
  "role" : "test-role",
  "resourceRole" : "test-role",
  "instanceType" : "m1.medium",
  "securityGroupIds" : "sg-12345678",
  "subnetId": "subnet-1a2b3c4d",
  "associatePublicIpAddress": "true",
  "keyPair" : "my-key-pair"
}
```

For more information, see Ec2Resource (p. 204).

Using Amazon EC2 Spot Instances in a Pipeline

Pipelines can use Amazon EC2 Spot Instances for the task nodes in their Amazon EMR cluster resources. By default, pipelines use on-demand Amazon EC2 instances. Spot Instances let you bid on spare Amazon EC2 instances and run them whenever your bid exceeds the current Spot Price, which varies in real-time based on supply and demand. The Spot Instance pricing model complements the on-demand and Reserved Instance pricing models, potentially providing the most cost-effective option for obtaining compute capacity, depending on your application. For more information, see Amazon EC2 Spot Instances on the *Amazon EC2 Product Page*.

To use Spot Instances in your pipeline

1. Open the AWS Data Pipeline console at https://console.aws.amazon.com/datapipeline/.
2. Open your pipeline in Architect.
3. In the **Resources** pane, go to the EMR cluster. In **Add an optional field**, select **Task Instance Bid Price**. Set **Task Instance Bid Price** to your Spot Instance bid price. This is the maximum dollar amount for your bid, and is a decimal value between 0 and 20.00 exclusive.

For more information, see EmrCluster (p. 210).

Using a Pipeline with Resources in Multiple Regions

By default, the `Ec2Resource` and `EmrCluster` resources run in the same region as AWS Data Pipeline, however AWS Data Pipeline supports the ability to orchestrate data flows across multiple regions, such as running resources in one region that consolidate input data from another region. By allowing resources to

run a specified region, you also have the flexibility to co-locate your resources with their dependent data sets and maximize performance by reducing latencies and avoiding cross-region data transfer charges. You can configure resources to run in a different region than AWS Data Pipeline by using the `region` field on `Ec2Resource` and `EmrCluster`.

The following example pipeline JSON file shows how to run an `EmrCluster` resource in the EU (Ireland) region, assuming that a large amount of data for the cluster to work on exists in the same region. In this example, the only difference from a typical pipeline is that the `EmrCluster` has a `region` field value set to `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"
    }
  ]
}
```

The following table lists the regions that you can choose and the associated region codes to use in the `region` field:

Region Name	Region Code
US East (N. Virginia)	us-east-1
US West (N. California)	us-west-1
US West (Oregon)	us-west-2
EU (Ireland)	eu-west-1
EU (Frankfurt)	eu-central-1
Asia Pacific (Singapore)	ap-southeast-1

Region Name	Region Code
Asia Pacific (Sydney)	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
South America (São Paulo)	sa-east-1

Cascading Failures and Reruns

AWS Data Pipeline allows you to configure the way pipeline objects behave when a dependency fails or is canceled by a user. You can ensure that failures cascade to other pipeline objects (consumers), to prevent indefinite waiting. All activities, data nodes, and preconditions have a field named `failureAndRerunMode` with a default value of `none`. To enable cascading failures, set the `failureAndRerunMode` field to `cascade`.

When this field is enabled, cascade failures occur if a pipeline object is blocked in the `WAITING_ON_DEPENDENCIES` state and any dependencies have failed with no pending command. During a cascade failure, the following events occur:

- When an object fails, its consumers are set to `CASCADE_FAILED` and both the original object and its consumers' preconditions are set to `CANCELED`.
- Any objects that are already `FINISHED`, `FAILED`, or `CANCELED` are ignored.

Cascade failure does not operate on a failed object's dependencies (upstream), except for preconditions associated with the original failed object. Pipeline objects affected by a cascade failure may trigger any retries or post-actions, such as `onFail`.

The detailed effects of a cascading failure depend on the object type.

Activities

An activity changes to `CASCADE_FAILED` if any of its dependencies fail, and it subsequently triggers a cascade failure in the activity's consumers. If a resource fails that the activity depends on, the activity is `CANCELED` and all its consumers change to `CASCADE_FAILED`.

Data Nodes and Preconditions

If a data node is configured as the output of an activity that fails, the data node changes to the `CASCADE_FAILED` state. The failure of a data node propagates to any associated preconditions, which change to the `CANCELED` state.

Resources

If the objects that depend on a resource are in the `FAILED` state and the resource itself is in the `WAITING_ON_DEPENDENCIES` state, then the resource changes to the `FINISHED` state.

Rerunning Cascade-Failed Objects

By default, rerunning any activity or data node only reruns the associated resource. However, setting the `failureAndRerunMode` field to `cascade` on a pipeline object allows a rerun command on a target object to propagate to all consumers, under the following conditions:

- The target object's consumers are in the CASCADE_FAILED state.
- The target object's dependencies have no rerun commands pending.
- The target object's dependencies are not in the FAILED, CASCADE_FAILED, or CANCELED state.

If you attempt to rerun a CASCADE_FAILED object and any of its dependencies are FAILED, CASCADE_FAILED, or CANCELED, the rerun will fail and return the object to the CASCADE_FAILED state. To successfully rerun the failed object, you must trace the failure up the dependency chain to locate the original source of failure and rerun that object instead. When you issue a rerun command on a resource, you also attempt to rerun any objects that depend on it.

Cascade-Failure and Backfills

If you enable cascade failure and have a pipeline that creates many backfills, pipeline runtime errors can cause resources to be created and deleted in rapid succession without performing useful work. AWS Data Pipeline attempts to alert you about this situation with the following warning message when you save a pipeline: *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. This happens because cascade failure can quickly set downstream activities as CASCADE_FAILED and shutdown EMR clusters and EC2 resources that are no longer needed. We recommended that you test pipelines with short time ranges to limit the effects of this situation.

Pipeline Definition File Syntax

The instructions in this section are for working manually with pipeline definition files using the AWS Data Pipeline command line interface (CLI). This is an alternative to designing a pipeline interactively using the AWS Data Pipeline console.

You can manually create pipeline definition files using any text editor that supports saving files using the UTF-8 file format and submit the files using the AWS Data Pipeline command line interface.

AWS Data Pipeline also supports a variety of complex expressions and functions within pipeline definitions. For more information, see Pipeline Expressions and Functions (p. 119).

File Structure

The first step in pipeline creation is to compose pipeline definition objects in a pipeline definition file. The following example illustrates the general structure of a pipeline definition file. This file defines two objects, which are delimited by '{' and '}', and separated by a comma.

In the following example, the first object defines two name-value pairs, known as *fields*. The second object defines three fields.

```
{
  "objects" : [
    {
       "name1" : "value1",
       "name2" : "value2"
    },
    {
       "name1" : "value3",
       "name3" : "value4",
       "name4" : "value5"
```

```
    }
  ]
}
```

When creating a pipeline definition file, you must select the types of pipeline objects that you'll need, add them to the pipeline definition file, and then add the appropriate fields. For more information about pipeline objects, see Pipeline Object Reference (p. 130).

For example, you could create a pipeline definition object for an input data node and another for the output data node. Then create another pipeline definition object for an activity, such as processing the input data using Amazon EMR.

Pipeline Fields

After you know which object types to include in your pipeline definition file, you add fields to the definition of each pipeline object. Field names are enclosed in quotes, and are separated from field values by a space, a colon, and a space, as shown in the following example.

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

The field value can be a text string, a reference to another object, a function call, an expression, or an ordered list of any of the preceding types. for more information about the types of data that can be used for field values, see Simple Data Types (p. 119) . For more information about functions that you can use to evaluate field values, see Expression Evaluation (p. 123).

Fields are limited to 2048 characters. Objects can be 20 KB in size, which means that you can't add many large fields to an object.

Each pipeline object must contain the following fields: `id` and `type`, as shown in the following example. Other fields may also be required based on the object type. Select a value for `id` that's meaningful to you, and is unique within the pipeline definition. The value for `type` specifies the type of the object. Specify one of the supported pipeline definition object types, which are listed in the topic Pipeline Object Reference (p. 130).

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

For more information about the required and optional fields for each object, see the documentation for the object.

To include fields from one object in another object, use the `parent` field with a reference to the object. For example, object "B" includes its fields, "B1" and "B2", plus the fields from object "A", "A1" and "A2".

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

You can define common fields in an object with the ID "Default". These fields are automatically included in every object in the pipeline definition file that doesn't explicitly set its `parent` field to reference a different object.

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

User-Defined Fields

You can create user-defined or custom fields on your pipeline components and refer to them with expressions. The following example shows a custom field named `myCustomField` and `my_customFieldReference` added to an S3DataNode object:

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

A user-defined field must have a name prefixed with the word "my" in all lower-case letters, followed by a capital letter or underscore character. Additionally, a user-defined field can be a string value such as the preceding `myCustomField` example, or a reference to another pipeline component such as the preceding `my_customFieldReference` example.

Note
On user-defined fields, AWS Data Pipeline only checks for valid references to other pipeline components, not any custom field string values that you add.

Working with the API

Note
If you are not writing programs that interact with AWS Data Pipeline, you do not need to install any of the AWS SDKs. You can create and run pipelines using the console or command-line interface. For more information, see Setting Up for AWS Data Pipeline (p. 10)

The easiest way to write applications that interact with AWS Data Pipeline or to implement a custom Task Runner is to use one of the AWS SDKs. The AWS SDKs provide functionality that simplify calling the web service APIs from your preferred programming environment. For more information, see Install the AWS SDK (p. 56).

Install the AWS SDK

The AWS SDKs provide functions that wrap the API and take care of many of the connection details, such as calculating signatures, handling request retries, and error handling. The SDKs also contain sample code, tutorials, and other resources to help you get started writing applications that call AWS. Calling the wrapper functions in an SDK can greatly simplify the process of writing an AWS application. For more information about how to download and use the AWS SDKs, go to Sample Code & Libraries.

AWS Data Pipeline support is available in SDKs for the following platforms:

- AWS SDK for Java
- AWS SDK for Node.js
- AWS SDK for PHP
- AWS SDK for Python (Boto)
- AWS SDK for Ruby
- AWS SDK for .NET

Making an HTTP Request to AWS Data Pipeline

For a complete description of the programmatic objects in AWS Data Pipeline, see the AWS Data Pipeline API Reference.

If you don't use one of the AWS SDKs, you can perform AWS Data Pipeline operations over HTTP using the POST request method. The POST method requires you to specify the operation in the header of the request and provide the data for the operation in JSON format in the body of the request.

HTTP Header Contents

AWS Data Pipeline requires the following information in the header of an HTTP request:

- `host` The AWS Data Pipeline endpoint.

  For information about endpoints, see Regions and Endpoints.
- `x-amz-date` You must provide the time stamp in either the HTTP Date header or the AWS x-amz-date header. (Some HTTP client libraries don't let you set the Date header.) When an x-amz-date header is present, the system ignores any Date header during the request authentication.

  The date must be specified in one of the following three formats, as specified in the HTTP/1.1 RFC:
  - Sun, 06 Nov 1994 08:49:37 GMT (RFC 822, updated by RFC 1123)
  - Sunday, 06-Nov-94 08:49:37 GMT (RFC 850, obsoleted by RFC 1036)
  - Sun Nov 6 08:49:37 1994 (ANSI C asctime() format)
- `Authorization` The set of authorization parameters that AWS uses to ensure the validity and authenticity of the request. For more information about constructing this header, go to Signature Version 4 Signing Process.
- `x-amz-target` The destination service of the request and the operation for the data, in the format: `<<serviceName>>_<<API version>>.<<operationName>>`

  For example, `DataPipeline_20121129.ActivatePipeline`
- `content-type` Specifies JSON and the version. For example, `Content-Type: application/x-amz-json-1.0`

The following is an example header for an HTTP request to activate a pipeline.

```
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 Body Content

The body of an HTTP request contains the data for the operation specified in the header of the HTTP request. The data must be formatted according to the JSON data schema for each AWS Data Pipeline API. The AWS Data Pipeline JSON data schema defines the types of data and parameters (such as comparison operators and enumeration constants) available for each operation.

Format the Body of an HTTP request

Use the JSON data format to convey data values and data structure, simultaneously. Elements can be nested within other elements by using bracket notation. The following example shows a request for putting a pipeline definition consisting of three objects and their corresponding slots.

```
{
 "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"}

      ]
    }
  ]
}
```

Handle the HTTP Response

Here are some important headers in the HTTP response, and how you should handle them in your application:

- **HTTP/1.1**—This header is followed by a status code. A code value of 200 indicates a successful operation. Any other value indicates an error.
- **x-amzn-RequestId**—This header contains a request ID that you can use if you need to troubleshoot a request with AWS Data Pipeline. An example of a request ID is K2QH8DNOU907N97FNA2GDLL8OBVV4KQNSO5AEMVJF66Q9ASUAAJG.
- **x-amz-crc32**—AWS Data Pipeline calculates a CRC32 checksum of the HTTP payload and returns this checksum in the x-amz-crc32 header. We recommend that you compute your own CRC32 checksum on the client side and compare it with the x-amz-crc32 header; if the checksums do not match, it might indicate that the data was corrupted in transit. If this happens, you should retry your request.

AWS SDK users do not need to manually perform this verification, because the SDKs compute the checksum of each reply from Amazon DynamoDB and automatically retry if a mismatch is detected.

Sample AWS Data Pipeline JSON Request and Response

The following examples show a request for creating a new pipeline. Then it shows the AWS Data Pipeline response, including the pipeline identifier of the newly created pipeline.

HTTP POST Request

```
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 Response

```
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"}
```

Controlling Access to Pipelines and Resources

Your security credentials identify you to services in AWS and grant you unlimited use of your AWS resources, such as your AWS Data Pipeline pipelines. You can use features of AWS Data Pipeline and AWS Identity and Access Management (IAM) to allow AWS Data Pipeline and other users to access your AWS Data Pipeline resources without sharing your security credentials.

Organizations can share access to pipelines so that the individuals in that organization can develop and maintain them collaboratively. However, for example, it might be necessary to do the following:

- Control which IAM users can access specific pipelines
- Protect a production pipeline from being edited by mistake
- Allow an auditor to have read-only access to pipelines, but prevent them from making changes

AWS Data Pipeline integrates with AWS Identity and Access Management (IAM), a service that enables you to do the following:

- Create users and groups under your AWS account
- Easily share your AWS resources between the users in your AWS account
- Assign unique security credentials to each user
- Control each user's access to services and resources
- Get a single bill for all users in your AWS account

By using IAM with AWS Data Pipeline, you can control whether users in your organization can perform a task using specific API actions and whether they can use specific AWS resources. You can use IAM policies based on pipeline tags and worker groups to share your pipelines with other users and control the level of access they have.

Contents

- IAM Roles for AWS Data Pipeline (p. 66)

IAM Policies for AWS Data Pipeline

By default, IAM users don't have permission to create or modify AWS resources. To allow IAM users to create or modify resources and perform tasks, you must create IAM policies that grant IAM users permission to use the specific resources and API actions they'll need, and then attach those policies to the IAM users or groups that require those permissions.

When you attach a policy to a user or group of users, it allows or denies the users permission to perform the specified tasks on the specified resources. For general information about IAM policies, see Permissions and Policies in the *IAM User Guide* guide. For more information about managing and creating custom IAM policies, see Managing IAM Policies.

Contents

Policy Syntax

An IAM policy is a JSON document that consists of one or more statements. Each statement is structured as follows:

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

The following elements make up a policy statement:

- **Effect:** The *effect* can be `Allow` or `Deny`. By default, IAM users don't have permission to use resources and API actions, so all requests are denied. An explicit allow overrides the default. An explicit deny overrides any allows.
- **Action**: The *action* is the specific API action for which you are granting or denying permission. For a list of actions for AWS Data Pipeline, see Actions in the *AWS Data Pipeline API Reference*.
- **Resource**: The resource that's affected by the action. The only valid value here is `"*"`.
- **Condition**: Conditions are optional. They can be used to control when your policy will be in effect.

  AWS Data Pipeline implements the AWS-wide context keys (see Available Keys for Conditions), plus the following service-specific keys.

  - `datapipeline:PipelineCreator` — To grant access to the user that created the pipeline. For an example, see Grant the pipeline owner full access (p. 65).
  - `datapipeline:Tag` — To grant access based on pipeline tagging. For more information, see Controlling Access to Pipelines Using Tags (p. 62).

- `datapipeline:workerGroup` — To grant access based on the name of the worker group. For more information, see Controlling Access to Pipelines Using Worker Groups (p. 63).

Controlling Access to Pipelines Using Tags

You can create IAM policies that reference the tags for your pipeline. This enables you to use pipeline tagging to do the following:

- Grant read-only access to a pipeline
- Grant read/write access to a pipeline
- Block access to a pipeline

For example, suppose that a manager has two pipeline environments, production and development, and an IAM group for each environment. For pipelines in the production environment, the manager grants read/write access to users in the production IAM group, but grants read-only access to users in the developer IAM group. For pipelines in the development environment, the manager grants read/write access to both the production and developer IAM groups.

To achieve this scenario, the manager tags the production pipelines with the "environment=production" tag and attaches the following policy to the developer IAM group. The first statement grants read-only access to all pipelines. The second statement grants read/write access to pipelines that do not have an "environment=production" tag.

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

In addition, the manager attaches the following policy to the production IAM group. This statement grants full access to all pipelines.

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

```
  ]
}
```

For more examples, see Grant users read-only access based on a tag (p. 64) and Grant users full access based on a tag (p. 65).

Controlling Access to Pipelines Using Worker Groups

You can create IAM policies that make reference worker group names.

For example, suppose that a manager has two pipeline environments, production and development, and an IAM group for each environment. The manager has three database servers with task runners configured for production, pre-production, and developer environments, respectively. The manager wants to ensure that users in the production IAM group can create pipelines that push tasks to production resources, and that users in the development IAM group can create pipelines that push tasks to both pre-production and developer resources.

To achieve this scenario, the manager installs task runner on the production resources with production credentials, and sets `workerGroup` to "prodresource". In addition, the manager installs task runner on the development resources with development credentials, and sets `workerGroup` to "pre-production" and "development". The manager attaches the following policy to the developer IAM group to block access to "prodresource" resources. The first statement grants read-only access to all pipelines. The second statement grants read/write access to pipelines when the name of the worker group has a prefix of "dev" or "pre-prod".

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

In addition, the manager attaches the following policy to the production IAM group to grant access to "prodresource" resources. The first statement grants read-only access to all pipelines. The second statement grants read/write access when the name of the worker group has a prefix of "prod".

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

Example Policies for AWS Data Pipeline

The following examples demonstrate how to grant users full or restricted access to pipelines.

Example 1: Grant users read-only access based on a tag

The following policy allows users to use the read-only AWS Data Pipeline API actions, but only with pipelines that have the tag "environment=production".

Note
The ListPipelines API action does not suppport tab-based authorization.

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

Example 2: Grant users full access based on a tag

The following policy allows users to use all AWS Data Pipeline API actions, with the exception of ListPipelines, but only with pipelines that have the tag "environment=test".

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

Example 3: Grant the pipeline owner full access

The following policy allows users to use all the AWS Data Pipeline API actions, but only with their own pipelines.

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

Example 4: Grant users access to the AWS Data Pipeline console

The following policy allows users to create and manage a pipeline using the AWS Data Pipeline console.

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

```
        "dynamodb:DescribeTable",
        "elasticmapreduce:AddJobFlowSteps",
        "elasticmapreduce:ListInstance*",
        "iam:AddRoleToInstanceProfile",
        "iam:CreateInstanceProfile",
        "iam:GetInstanceProfile",
        "iam:GetRole",
        "iam:ListInstanceProfiles",
        "iam:ListInstanceProfilesForRole",
        "iam:ListRoles",
        "iam:PassRole",
        "rds:DescribeDBInstances",
        "rds:DescribeDBSecurityGroups",
        "redshift:DescribeClusters",
        "redshift:DescribeClusterSecurityGroups",
        "s3:List*",
        "sns:ListTopics"
      ],
      "Resource": "*"
    }
  ]
}
```

IAM Roles for AWS Data Pipeline

AWS Data Pipeline requires IAM roles to determine what actions your pipelines can perform and what resources it can access. Additionally, when a pipeline creates a resource, such as an EC2 instance or EMR cluster, IAM roles determine what actions your applications can perform and what resources they can access.

The AWS Data Pipeline console creates the following roles for you:

- **DataPipelineDefaultRole** - Grants AWS Data Pipeline access to your AWS resources
- **DataPipelineDefaultResourceRole** - Grants your applications access to your AWS resources

If you are using a CLI or an API and you have not used the AWS Data Pipeline console to create a pipeline previously, you must create these roles manually using AWS Identity and Access Management (IAM). For more information, see Create the Required IAM Roles (for CLI or API only) (p. 10).

Alternatively, you can create custom roles. For an example of how to specify these roles for an `EmrCluster` object, see Specify custom IAM roles (p. 212).

Update Existing IAM Roles for AWS Data Pipeline

If the **DataPipelineDefaultRole** and **DataPipelineDefaultResourceRole** roles were created using inline policies instead of managed policies, the AWS account owner can update them to use managed policies. After you update your roles to use an AWS managed policy, they will receive future updates automatically.

Use the following procedure to update the **DataPipelineDefaultRole** and **DataPipelineDefaultResourceRole** roles.

To update your existing IAM roles using managed policies

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. Update the **DataPipelineDefaultRole** role as follows:

    a. In the navigation pane, click **Roles**, and then click the row for the **DataPipelineDefaultRole** role.

b. Under **Permissions**, click **Remove Policy** for the inline policy. When prompted for confirmation, click **Remove**.
c. Under **Permissions**, click **Attach Policy**.
d. On the **Attach Policy** page, click the box next to the **AWSDataPipelineRole** policy, and then click **Attach Policy**.

3. Update the **DataPipelineDefaultResourceRole** role as follows:

a. In the navigation pane, click **Roles**, and then click the row for the **DataPipelineDefaultResourceRole** role
b. Under **Permissions**, click **Remove Policy** for the inline policy. When prompted for confirmation, click **Remove**.
c. Under **Permissions**, click **Attach Policy**.
d. On the **Attach Policy** page, click the box next to the **AmazonEC2RoleforDataPipelineRole** policy, and then click **Attach Policy**.

If you prefer to maintain the inline policy yourself, you can do so as follows.

To update your existing IAM roles using inline policies

1. Update `DataPipelineDefaultRole` to use the following policy:

```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Effect": "Allow",
      "Action": [
        "cloudwatch:*",
        "datapipeline:DescribeObjects",
        "datapipeline:EvaluateExpression",
        "dynamodb:BatchGetItem",
        "dynamodb:DescribeTable",
        "dynamodb:GetItem",
        "dynamodb:Query",
        "dynamodb:Scan",
        "dynamodb:UpdateTable",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:CancelSpotInstanceRequests",
        "ec2:CreateSecurityGroup",
        "ec2:CreateTags",
        "ec2:DeleteTags",
        "ec2:Describe*",
        "ec2:ModifyImageAttribute",
        "ec2:ModifyInstanceAttribute",
        "ec2:RequestSpotInstances",
        "ec2:RunInstances",
        "ec2:StartInstances",
        "ec2:StopInstances",
        "ec2:TerminateInstances",
        "elasticmapreduce:*",
        "iam:GetInstanceProfile",
        "iam:GetRole",
        "iam:GetRolePolicy",
        "iam:ListAttachedRolePolicies",
        "iam:ListInstanceProfiles",
        "iam:ListRolePolicies",
        "iam:PassRole",
        "rds:DescribeDBInstances",
        "rds:DescribeDBSecurityGroups",
        "redshift:DescribeClusters",
        "redshift:DescribeClusterSecurityGroups",
```

```
        "s3:CreateBucket",
        "s3:DeleteObject",
        "s3:Get*",
        "s3:List*",
        "s3:Put*",
        "sdb:BatchPutAttributes",
        "sdb:Select*",
        "sns:GetTopicAttributes",
        "sns:ListTopics",
        "sns:Publish",
        "sns:Subscribe",
        "sns:Unsubscribe"
      ],
      "Resource": ["*"]
    }]
}
```

2. Update `DataPipelineDefaultRole` to use the following trusted entities list:

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

3. Update `DataPipelineDefaultResourceRole` to use the following policy:

```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Effect": "Allow",
      "Action": [
        "cloudwatch:*",
        "datapipeline:*",
        "dynamodb:*",
        "ec2:Describe*",
        "elasticmapreduce:AddJobFlowSteps",
        "elasticmapreduce:Describe*",
        "elasticmapreduce:ListInstance*",
        "rds:Describe*",
        "redshift:DescribeClusters",
        "redshift:DescribeClusterSecurityGroups",
        "s3:*",
        "sdb:*",
        "sns:*",
        "sqs:*"
      ],
      "Resource": ["*"]
    }]
}
```

4. Update `DataPipelineDefaultResourceRole` to use the following trusted entities list:

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "ec2.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Change Roles on Existing Pipelines

If you have a custom role and you would like to change roles for existing pipelines by editing the pipeline in the AWS Data Pipeline console:

1. Open the AWS Data Pipeline console at https://console.aws.amazon.com/datapipeline/.
2. Select the pipeline you desire to edit by clicking the pipeline ID.
3. Choose **Edit Pipeline**.
4. Choose the **Others** dropdown and enter the appropriate **Role** and **Resource Role**.

Tutorials

The following tutorials walk you step-by-step through the process of creating and using pipelines with AWS Data Pipeline.

Tutorials

- Process Data Using Amazon EMR with Hadoop Streaming (p. 70)
- Import and Export DynamoDB Data Using AWS Data Pipeline (p. 77)
- Copy CSV Data Between Amazon S3 Buckets Using AWS Data Pipeline (p. 86)
- Export MySQL Data to Amazon S3 Using AWS Data Pipeline (p. 96)
- Copy Data to Amazon Redshift Using AWS Data Pipeline (p. 107)

Process Data Using Amazon EMR with Hadoop Streaming

You can use AWS Data Pipeline to manage your Amazon EMR clusters. With AWS Data Pipeline you can specify preconditions that must be met before the cluster is launched (for example, ensuring that today's data been uploaded to Amazon S3), a schedule for repeatedly running the cluster, and the cluster configuration to use. The following tutorial walks you through launching a simple cluster.

In this tutorial, you create a pipeline for a simple Amazon EMR cluster to run a pre-existing Hadoop Streaming job provided by Amazon EMR and send an Amazon SNS notification after the task completes successfully. You use the Amazon EMR cluster resource provided by AWS Data Pipeline for this task. The sample application is called WordCount, and can also be run manually from the Amazon EMR console. Note that clusters spawned by AWS Data Pipeline on your behalf are displayed in the Amazon EMR console and are billed to your AWS account.

Pipeline Objects

The pipeline uses the following objects:

EmrActivity (p. 156)

Defines the work to perform in the pipeline (run a pre-existing Hadoop Streaming job provided by Amazon EMR).

EmrCluster (p. 210)

Resource AWS Data Pipeline uses to perform this activity.

A cluster is a set of Amazon EC2 instances. AWS Data Pipeline launches the cluster and then terminates it after the task finishes.

Schedule (p. 252)

Start date, time, and the duration for this activity. You can optionally specify the end date and time.

SnsAlarm (p. 250)

Sends an Amazon SNS notification to the topic you specify after the task finishes successfully.

Contents

Before You Begin

Be sure you've completed the following steps.

- Complete the tasks in Setting Up for AWS Data Pipeline (p. 10).
- (Optional) Set up a VPC for the cluster and a security group for the VPC. For more information, see Launching Resources for Your Pipeline into a VPC (p. 48).
- Create a topic for sending email notification and make a note of the topic Amazon Resource Name (ARN). For more information, see Create a Topic in the *Amazon Simple Notification Service Getting Started Guide*.

Launch a Cluster Using the AWS Data Pipeline Console

You can create a pipeline to launch a cluster to analyze web logs or perform analysis of scientific data.

Tasks

Create the Pipeline

First, create the pipeline.

To create your pipeline

1. Open the AWS Data Pipeline console at https://console.aws.amazon.com/datapipeline/.
2. The first screen that you see depends on whether you've created a pipeline in the current region.

   a. If you haven't created a pipeline in this region, the console displays an introductory screen. Choose **Get started now**.
   b. If you've already created a pipeline in this region, the console displays a page that lists your pipelines for the region. Choose **Create new pipeline**.
3. In **Name**, enter a name for your pipeline.
4. (Optional) In **Description**, enter a description for your pipeline.
5. For **Source**, select **Build using Architect**.
6. Under **Schedule**, choose **on pipeline activation**.
7. Under **Pipeline Configuration**, leave logging enabled. Choose the folder icon under **S3 location for logs**, select one of your buckets or folders, and then choose **Select**.

   If you prefer, you can disable logging instead.
8. Under **Security/Access**, leave **IAM roles** set to **Default**.
9. Click **Edit in Architect**.

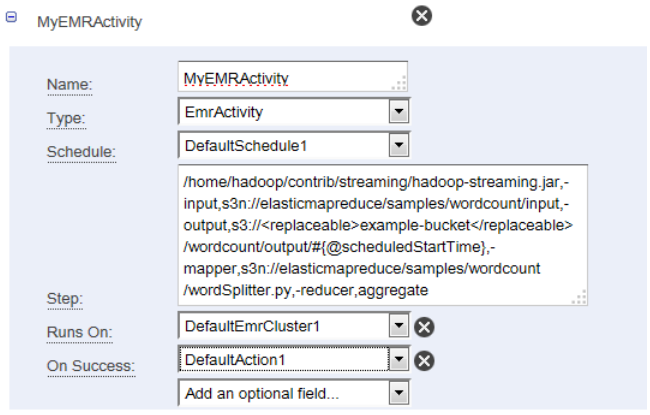
Next, add an activity to your pipeline definition. This also defines the other objects that AWS Data Pipeline must use to perform this activity.

To configure the activity

1. Click **Add activity**.
2. In the **Activities** pane:
   a. In the **Name** field, enter the name for the activity (for example, `MyEMRActivity`).
   b. From **Type**, select **EmrActivity**.
   c. In **Step**, enter:

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

   d. In **Add an optional field**, select **Runs On**. Set the value to **Create new: EmrCluster**.
   e. In **Add an optional field**, select **On Success**. Set the value to **Create new: Action**.



Next, configure the resource that AWS Data Pipeline uses to perform the Amazon EMR job.

To configure the resource

1. In the right pane, choose **Resources**.
2. In the **Name** field, enter the name for your Amazon EMR cluster (for example, `MyEMRCluster`).
3. Leave **Type** set to **EmrCluster**.
4. [EC2-VPC] (Optional) From **Add an optional field**, select **Subnet Id**. Set the value to the ID of the subnet.
5. (Optional) From **Add an optional field**, select **Enable Debugging**. Set the value to **`true`**.

   Note
   This option can incur extra costs because of log data storage. Use this option selectively, such as for prototyping and troubleshooting.
6. (Optional) In the right pane, choose **Others**. Under `Default`, from **Add an optional field**, select **Pipeline Log Uri**. Set the value to an Amazon S3 bucket for the Amazon EMR logs. For example, **`s3://examples-bucket/emrlogs`**.

   Note
   This option can incur extra costs because of log file storage. Use this option selectively, such as for prototyping and troubleshooting.

Next, configure the Amazon SNS notification action that AWS Data Pipeline performs after the Amazon EMR job finishes successfully.

To configure the notification action

1. In the right pane, click **Others**.
2. Under `DefaultAction1`, do the following:
   a. Enter the name for your notification (for example, `MyEMRJobNotice`).
   b. From **Type**, select **SnsAlarm**.
   c. In the **Subject** field, enter the subject line for your notification.
   d. In the **Topic Arn** field, enter the ARN of your topic (see Create a Topic).
   e. In **Message**, enter the message content.
   f. Leave **Role** set to the default value.

Save and Validate Your Pipeline

You can save your pipeline definition at any point during the creation process. As soon as you save your pipeline definition, AWS Data Pipeline looks for syntax errors and missing values in your pipeline definition. If your pipeline is incomplete or incorrect, AWS Data Pipeline generates validation errors and warnings. Warning messages are informational only, but you must fix any error messages before you can activate your pipeline.

To save and validate your pipeline

1. Choose **Save pipeline**.
2. AWS Data Pipeline validates your pipeline definition and returns either success or error or warning messages. If you get an error message, choose **Close** and then, in the right pane, choose **Errors/Warnings**.
3. The **Errors/Warnings** pane lists the objects that failed validation. Choose the plus (**+**) sign next to the object names and look for an error message in red.
4. When you see an error message, go to the specific object pane where you see the error and fix it. For example, if you see an error message in the **DataNodes** object, go to the **DataNodes** pane to fix the error.

5. After you fix the errors listed in the **Errors/Warnings** pane, choose **Save Pipeline**.
6. Repeat the process until your pipeline validates successfully.

Activate Your Pipeline

Activate your pipeline to start creating and processing runs. The pipeline starts based on the schedule and period in your pipeline definition.

> **Important**
> If activation succeeds, your pipeline is running and might incur usage charges. For more information, see AWS Data Pipeline pricing. To stop incurring usage charges for AWS Data Pipeline, delete your pipeline.

To activate your pipeline

1. Choose **Activate**.
2. In the confirmation dialog box, choose **Close**.

Monitor the Pipeline Runs

After you activate your pipeline, you are taken to the **Execution details** page where you can monitor the progress of your pipeline.

To monitor the progress of your pipeline runs

1. Choose **Update** or press F5 to update the status displayed.

   > **Tip**
   > If there are no runs listed, ensure that **Start (in UTC)** and **End (in UTC)** cover the scheduled start and end of your pipeline, and then choose **Update**.
2. When the status of every object in your pipeline is `FINISHED`, your pipeline has successfully completed the scheduled tasks. If you created an SNS notification, you should receive email about the successful completion of this task.
3. If your pipeline doesn't complete successfully, check your pipeline settings for issues. For more information about troubleshooting failed or incomplete instance runs of your pipeline, see Resolving Common Problems (p. 273).

(Optional) Delete Your Pipeline

To stop incurring charges, delete your pipeline. Deleting your pipeline deletes the pipeline definition and all associated objects.

To delete your pipeline

1. On the **List Pipelines** page, select your pipeline.
2. Click **Actions**, and then choose **Delete**.
3. When prompted for confirmation, choose **Delete**.

Launch a Cluster Using the Command Line

If you regularly run an Amazon EMR cluster to analyze web logs or perform analysis of scientific data, you can use AWS Data Pipeline to manage your Amazon EMR clusters. With AWS Data Pipeline, you can specify preconditions that must be met before the cluster is launched (for example, ensuring that today's

data been uploaded to Amazon S3.) This tutorial walks you through launching a cluster that can be a model for a simple Amazon EMR-based pipeline, or as part of a more involved pipeline.

Prerequisites

Before you can use the CLI, you must complete the following steps:

1. Install and configure a command line interface (CLI). For more information, see Accessing AWS Data Pipeline (p. 2).
2. Ensure that the IAM roles named **DataPipelineDefaultRole** and **DataPipelineDefaultResourceRole** exist. The AWS Data Pipeline console creates these roles for you automatically. If you haven't used the AWS Data Pipeline console at least once, then you must create these roles manually. For more information, see IAM Roles for AWS Data Pipeline (p. 66).

Tasks

- Creating the Pipeline Definition File (p. 75)
- Uploading and Activating the Pipeline Definition (p. 76)
- Monitor the Pipeline Runs (p. 77)

Creating the Pipeline Definition File

The following code is the pipeline definition file for a simple Amazon EMR cluster that runs an existing Hadoop streaming job provided by Amazon EMR. This sample application is called WordCount, and you can also run it using the Amazon EMR console.

Copy this code into a text file and save it as `MyEmrPipelineDefinition.json`. You should replace the Amazon S3 bucket location with the name of an Amazon S3 bucket that you own. You should also replace the start and end dates. To launch clusters immediately, set `startDateTime` to a date one day in the past and `endDateTime` to one day in the future. AWS Data Pipeline then starts launching the "past due" clusters immediately in an attempt to address what it perceives as a backlog of work. This backfilling means you don't have to wait an hour to see AWS Data Pipeline launch its first cluster.

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

This pipeline has three objects:

- `Hourly`, which represents the schedule of the work. You can set a schedule as one of the fields on an activity. When you do, the activity runs according to that schedule, or in this case, hourly.
- `MyCluster`, which represents the set of Amazon EC2 instances used to run the cluster. You can specify the size and number of EC2 instances to run as the cluster. If you do not specify the number of instances, the cluster launches with two, a master node and a task node. You can specify a subnet to launch the cluster into. You can add additional configurations to the cluster, such as bootstrap actions to load additional software onto the Amazon EMR-provided AMI.
- `MyEmrActivity`, which represents the computation to process with the cluster. Amazon EMR supports several types of clusters, including streaming, Cascading, and Scripted Hive. The `runsOn` field refers back to MyCluster, using that as the specification for the underpinnings of the cluster.

Uploading and Activating the Pipeline Definition

You must upload your pipeline definition and activate your pipeline. In the following example commands, replace *`pipeline_name`* with a label for your pipeline and *`pipeline_file`* with the fully-qualified path for the pipeline definition `.json` file.

AWS CLI

To create your pipeline definition and activate your pipeline, use the following create-pipeline command. Note the ID of your pipeline, because you'll use this value with most CLI commands.

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

To upload your pipeline definition, use the following put-pipeline-definition command.

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

If you pipeline validates successfully, the `validationErrors` field is empty. You should review any warnings.

To activate your pipeline, use the following activate-pipeline command.

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

You can verify that your pipeline appears in the pipeline list using the following list-pipelines command.

```
aws datapipeline list-pipelines
```

AWS Data Pipeline CLI

To upload your pipeline definition and activate your pipeline in a single step, use the following command.

```
datapipeline --create pipeline_name --put pipeline_file --activate --force
```

If your pipeline validates successfully, the command displays the following message. Note the ID of your pipeline, because you'll use this value with most AWS Data Pipeline CLI commands.

```
Pipeline with name pipeline_name and id pipeline_id created.
Pipeline definition pipeline_file uploaded.
Pipeline activated.
```

If the command fails, you'll see an error message. For information, see Troubleshooting (p. 270).

You can verify that your pipeline appears in the pipeline list using the following command.

```
datapipeline --list-pipelines
```

Monitor the Pipeline Runs

You can view clusters launched by AWS Data Pipeline using the Amazon EMR console and you can view the output folder using the Amazon S3 console.

To check the progress of clusters launched by AWS Data Pipeline

1. Open the Amazon EMR console.
2. The clusters that were spawned by AWS Data Pipeline have a name formatted as follows: *<pipeline-identifier>*_@*<emr-cluster-name>*_*<launch-time>*.
3. After one of the runs is complete, open the Amazon S3 console and check that the time-stamped output folder exists and contains the expected results of the cluster.

Import and Export DynamoDB Data Using AWS Data Pipeline

These tutorials demonstrate how to move schema-less data in and out of Amazon DynamoDB using AWS Data Pipeline, which in turn employs Amazon EMR and Hive. Complete part one before you move on to part two.

Tutorials

- Part One: Import Data into DynamoDB (p. 77)
- Part Two: Export Data from DynamoDB (p. 82)

Part One: Import Data into DynamoDB

The first part of this tutorial explains how to define an AWS Data Pipeline pipeline to retrieve data from a tab-delimited file in Amazon S3 to populate a DynamoDB table, use a Hive script to define the necessary data transformation steps, and automatically create an Amazon EMR cluster to perform the work.

Tasks

- Before You Begin (p. 78)
- Step 1: Create the Pipeline (p. 79)
- Step 2: Save and Validate Your Pipeline (p. 80)
- Step 3: Activate Your Pipeline (p. 80)

Before You Begin

Be sure to complete the following steps:

- Complete the tasks in Setting Up for AWS Data Pipeline (p. 10).
- (Optional) Set up a VPC for the cluster and a security group for the VPC. For more information, see Launching Resources for Your Pipeline into a VPC (p. 48).
- Create a topic and subscribe to receive notifications from AWS Data Pipeline regarding the status of your pipeline components. For more information, see Create a Topic in the *Amazon Simple Notification Service Getting Started Guide*.
- Create a DynamoDB table to store data. For more information, see Create a DynamoDB Table (p. 78).

Be aware of the following:

- Imports may overwrite data in your DynamoDB table. When you import data from Amazon S3, the import may overwrite items in your DynamoDB table. Make sure that you are importing the right data and into the right table. Be careful not to accidentally set up a recurring import pipeline that will import the same data multiple times.
- Exports may overwrite data in your Amazon S3 bucket. When you export data to Amazon S3, you may overwrite previous exports if you write to the same bucket path. The default behavior of the **Export DynamoDB to S3** template will append the job's scheduled time to the Amazon S3 bucket path, which will help you avoid this problem.
- Import and Export jobs will consume some of your DynamoDB table's provisioned throughput capacity. This section explains how to schedule an import or export job using Amazon EMR. The Amazon EMR cluster will consume some read capacity during exports or write capacity during imports. You can control the percentage of the provisioned capacity that the import/export jobs consume by with the settings `MyImportJob.myDynamoDBWriteThroughputRatio` and `MyExportJob.myDynamoDBReadThroughputRatio`. Be aware that these settings determine how much capacity to consume at the beginning of the import/export process and will not adapt in real time if you change your table's provisioned capacity in the middle of the process.
- Be aware of the costs. AWS Data Pipeline manages the import/export process for you, but you still pay for the underlying AWS services that are being used. The import and export pipelines will create Amazon EMR clusters to read and write data and there are per-instance charges for each node in the cluster. You can read more about the details of Amazon EMR Pricing. The default cluster configuration is one m1.small instance master node and one m1.xlarge instance task node, though you can change this configuration in the pipeline definition. There are also charges for AWS Data Pipeline. For more information, see AWS Data Pipeline Pricing and Amazon S3 Pricing.

Create a DynamoDB Table

You can create the DynamoDB table that is required for this tutorial. If you already have a DynamoDB table, you can skip this procedure to create one.

For more information, see Working with Tables in DynamoDB in the *Amazon DynamoDB Developer Guide*.

To create a DynamoDB table

1. Open the DynamoDB console at https://console.aws.amazon.com/dynamodb/.

2. Click **Create Table**.
3. Under **Primary Key**, enter a table name.
4. Enter a unique name for your table in **Table Name**.
5. In the **Primary Key : Partition Key** field, enter the string `Id`.
6. Click **Continue** to skip the optional **Add Indexes** page.
7. On the **Provisioned Throughput Capacity** page, do the following. Note that these values are small because the sample data is small. For information about calculating the required size for your own data, see Provisioned Throughput in Amazon DynamoDB in the *Amazon DynamoDB Developer Guide*.

   a. In **Read Capacity Units**, enter `5`.
   b. In **Write Capacity Units**, enter `5`.
   c. Click **Continue**.
8. On the **Throughput Alarms** page, in **Send notification to**, enter your email address, and then click **Continue**.
9. On the **Review** page, click **Create**.

Step 1: Create the Pipeline

First, create the pipeline.

To create the pipeline

1. Open the AWS Data Pipeline console at https://console.aws.amazon.com/datapipeline/.
2. The first screen that you see depends on whether you've created a pipeline in the current region.

   a. If you haven't created a pipeline in this region, the console displays an introductory screen. Choose **Get started now**.
   b. If you've already created a pipeline in this region, the console displays a page that lists your pipelines for the region. Choose **Create new pipeline**.
3. In **Name**, enter a name for your pipeline.
4. (Optional) In **Description**, enter a description for your pipeline.
5. For **Source**, select **Build using a template**, and then select the following template: **Import DynamoDB backup data from S3**.
6. Under **Parameters**, set **Input S3 folder** to `s3://elasticmapreduce/samples/Store/ProductCatalog`, which is a sample data source, and set **DynamoDB table name** to the name of your table.
7. Under **Schedule**, choose **on pipeline activation**.
8. Under **Pipeline Configuration**, leave logging enabled. Choose the folder icon under **S3 location for logs**, select one of your buckets or folders, and then choose **Select**.

   If you prefer, you can disable logging instead.
9. Under **Security/Access**, leave **IAM roles** set to **Default**.
10. Click **Edit in Architect**.

Next, configure the Amazon SNS notification actions that AWS Data Pipeline performs depending on the outcome of the activity.

To configure the success and failure actions

1. In the right pane, click **Activities**.

2. From **Add an optional field**, select **On Success**.
3. From the newly added **On Success**, select **Create new: Action**.
4. From **Add an optional field**, select **On Fail**.
5. From the newly added **On Fail**, select **Create new: Action**.
6. In the right pane, click **Others**.
7. For `DefaultAction1`, do the following:
   a. Change the name to `SuccessSnsAlarm`.
   b. From **Type**, select `SnsAlarm`.
   c. In **Topic Arn**, enter the ARN of the topic that you created (see ARN resource names for Amazon SNS.
   d. Enter a subject and a message.
8. For `DefaultAction2`, do the following:
   a. Change the name to `FailureSnsAlarm`.
   b. From **Type**, select `SnsAlarm`.
   c. In **Topic Arn**, enter the ARN of the topic that you created (see ARN resource names for Amazon SNS.
   d. Enter a subject and a message.

Step 2: Save and Validate Your Pipeline

You can save your pipeline definition at any point during the creation process. As soon as you save your pipeline definition, AWS Data Pipeline looks for syntax errors and missing values in your pipeline definition. If your pipeline is incomplete or incorrect, AWS Data Pipeline generates validation errors and warnings. Warning messages are informational only, but you must fix any error messages before you can activate your pipeline.

To save and validate your pipeline

1. Choose **Save pipeline**.
2. AWS Data Pipeline validates your pipeline definition and returns either success or error or warning messages. If you get an error message, choose **Close** and then, in the right pane, choose **Errors/Warnings**.
3. The **Errors/Warnings** pane lists the objects that failed validation. Choose the plus (**+**) sign next to the object names and look for an error message in red.
4. When you see an error message, go to the specific object pane where you see the error and fix it. For example, if you see an error message in the **DataNodes** object, go to the **DataNodes** pane to fix the error.
5. After you fix the errors listed in the **Errors/Warnings** pane, choose **Save Pipeline**.
6. Repeat the process until your pipeline validates successfully.

Step 3: Activate Your Pipeline

Activate your pipeline to start creating and processing runs. The pipeline starts based on the schedule and period in your pipeline definition.

> **Important**
> If activation succeeds, your pipeline is running and might incur usage charges. For more information, see AWS Data Pipeline pricing. To stop incurring usage charges for AWS Data Pipeline, delete your pipeline.

To activate your pipeline

1. Choose **Activate**.
2. In the confirmation dialog box, choose **Close**.

Step 4: Monitor the Pipeline Runs

After you activate your pipeline, you are taken to the **Execution details** page where you can monitor the progress of your pipeline.

To monitor the progress of your pipeline runs

1. Choose **Update** or press F5 to update the status displayed.

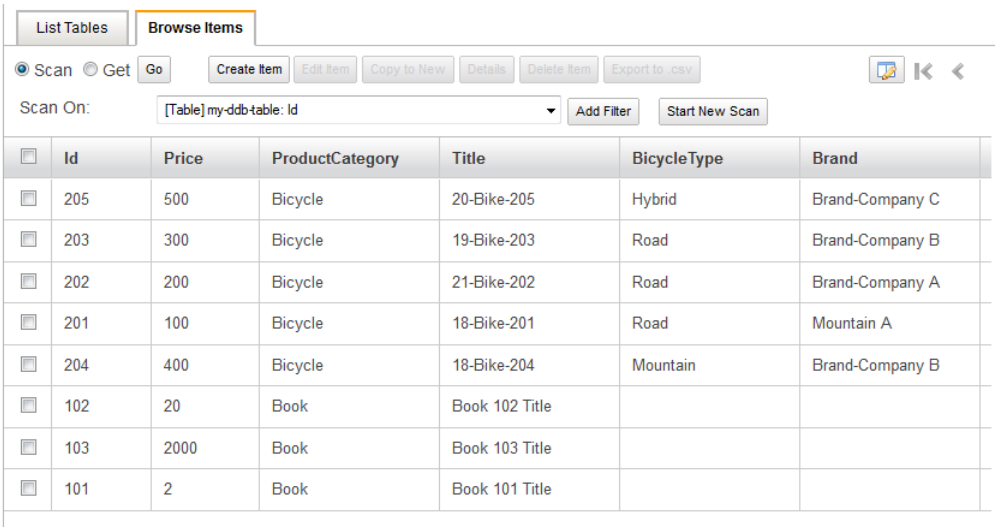
   Tip
   If there are no runs listed, ensure that **Start (in UTC)** and **End (in UTC)** cover the scheduled start and end of your pipeline, and then choose **Update**.
2. When the status of every object in your pipeline is `FINISHED`, your pipeline has successfully completed the scheduled tasks. If you created an SNS notification, you should receive email about the successful completion of this task.
3. If your pipeline doesn't complete successfully, check your pipeline settings for issues. For more information about troubleshooting failed or incomplete instance runs of your pipeline, see Resolving Common Problems (p. 273).

Step 5: Verify the Data Import

Next, verify that the data import occurred successfully using the DynamoDB console to inspect the data in the table.

To verify the DynamoDB table

1. Open the DynamoDB console.
2. On the **Tables** screen, click your DynamoDB table and click **Explore Table**.
3. On the **Browse Items** tab, columns that correspond to the data input file should display, such as Id, Price, ProductCategory, as shown in the following screen. This indicates that the import operation from the file to the DynamoDB table occurred successfully.



Id	Price	ProductCategory	Title	BicycleType	Brand
205	500	Bicycle	20-Bike-205	Hybrid	Brand-Company C
203	300	Bicycle	19-Bike-203	Road	Brand-Company B
202	200	Bicycle	21-Bike-202	Road	Brand-Company A
201	100	Bicycle	18-Bike-201	Road	Mountain A
204	400	Bicycle	18-Bike-204	Mountain	Brand-Company B
102	20	Book	Book 102 Title		
103	2000	Book	Book 103 Title		
101	2	Book	Book 101 Title		

Step 6: Delete Your Pipeline (Optional)

To stop incurring charges, delete your pipeline. Deleting your pipeline deletes the pipeline definition and all associated objects.

To delete your pipeline

1. On the **List Pipelines** page, select your pipeline.
2. Click **Actions**, and then choose **Delete**.
3. When prompted for confirmation, choose **Delete**.

Part Two: Export Data from DynamoDB

This is the second of a two-part tutorial that demonstrates how to bring together multiple AWS features to solve real-world problems in a scalable way through a common scenario: moving schema-less data in and out of DynamoDB using AWS Data Pipeline, which in turn employs Amazon EMR and Hive.

Tasks

Before You Begin

You must complete part one of this tutorial to ensure that your DynamoDB table contains the necessary data to perform the steps in this section. For more information, see Part One: Import Data into DynamoDB (p. 77).

Additionally, be sure you've completed the following steps:

- Complete the tasks in Setting Up for AWS Data Pipeline (p. 10).
- Create a topic and subscribe to receive notifications from AWS Data Pipeline regarding the status of your pipeline components. For more information, see Create a Topic in the *Amazon SNS Getting Started Guide*.
- Ensure that you have the DynamoDB table that was created and populated with data in part one of this tutorial. This table will be your data source for part two of the tutorial. For more information, see Part One: Import Data into DynamoDB (p. 77).

Be aware of the following:

- Imports may overwrite data in your DynamoDB table. When you import data from Amazon S3, the import may overwrite items in your DynamoDB table. Make sure that you are importing the right data and into the right table. Be careful not to accidentally set up a recurring import pipeline that will import the same data multiple times.
- Exports may overwrite data in your Amazon S3 bucket. When you export data to Amazon S3, you may overwrite previous exports if you write to the same bucket path. The default behavior of the **Export**

DynamoDB to S3 template will append the job's scheduled time to the Amazon S3 bucket path, which will help you avoid this problem.

- Import and Export jobs will consume some of your DynamoDB table's provisioned throughput capacity. This section explains how to schedule an import or export job using Amazon EMR. The Amazon EMR cluster will consume some read capacity during exports or write capacity during imports. You can control the percentage of the provisioned capacity that the import/export jobs consume by with the settings `MyImportJob.myDynamoDBWriteThroughputRatio` and `MyExportJob.myDynamoDBReadThroughputRatio`. Be aware that these settings determine how much capacity to consume at the beginning of the import/export process and will not adapt in real time if you change your table's provisioned capacity in the middle of the process.
- Be aware of the costs. AWS Data Pipeline manages the import/export process for you, but you still pay for the underlying AWS services that are being used. The import and export pipelines will create Amazon EMR clusters to read and write data and there are per-instance charges for each node in the cluster. You can read more about the details of Amazon EMR Pricing. The default cluster configuration is one m1.small instance master node and one m1.xlarge instance task node, though you can change this configuration in the pipeline definition. There are also charges for AWS Data Pipeline. For more information, see AWS Data Pipeline Pricing and Amazon S3 Pricing.

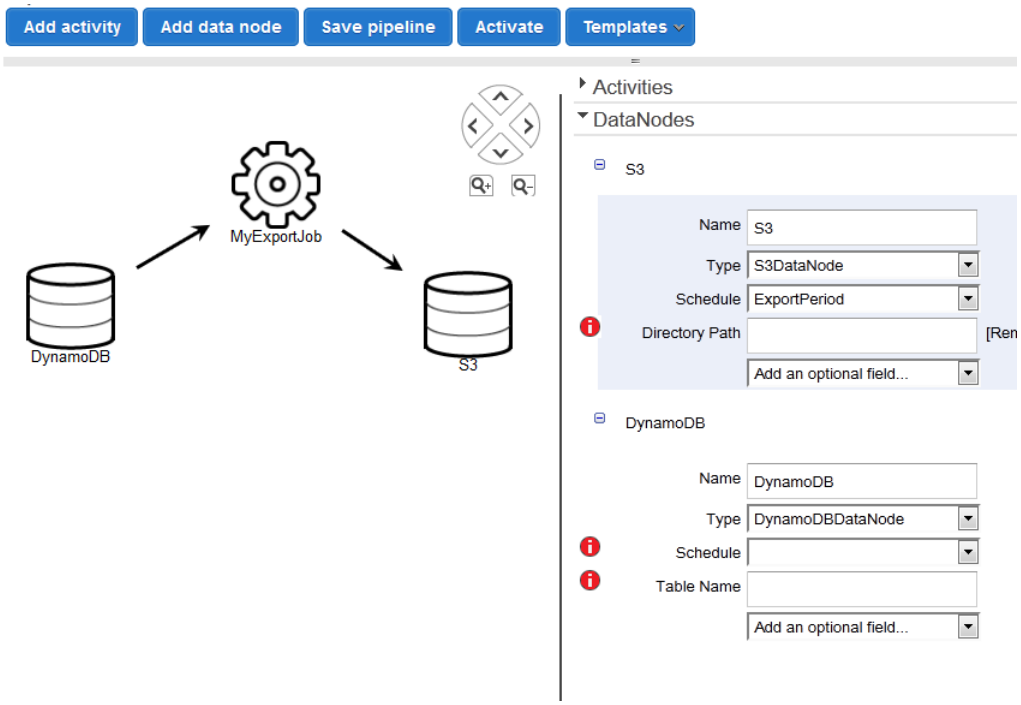
Step 1: Create the Pipeline

First, create the pipeline.

To create the pipeline

1. Open the AWS Data Pipeline console at https://console.aws.amazon.com/datapipeline/.
2. The first screen that you see depends on whether you've created a pipeline in the current region.
   a. If you haven't created a pipeline in this region, the console displays an introductory screen. Choose **Get started now**.
   b. If you've already created a pipeline in this region, the console displays a page that lists your pipelines for the region. Choose **Create new pipeline**.
3. In **Name**, enter a name for your pipeline.
4. (Optional) In **Description**, enter a description for your pipeline.
5. For **Source**, select **Build using a template**, and then select the following template: **Export DynamoDB table to S3**.
6. Under **Parameters**, set **DynamoDB table name** to the name of your table. Click the folder icon next to **Output S3 folder**, select one of your Amazon S3 buckets, and then click **Select**.
7. Under **Schedule**, choose **on pipeline activation**.
8. Under **Pipeline Configuration**, leave logging enabled. Choose the folder icon under **S3 location for logs**, select one of your buckets or folders, and then choose **Select**.

   If you prefer, you can disable logging instead.
9. Under **Security/Access**, leave **IAM roles** set to **Default**.
10. Click **Edit in Architect**.



Next, configure the Amazon SNS notification actions that AWS Data Pipeline performs depending on the outcome of the activity.

To configure the success, failure, and late notification actions

1. In the right pane, click **Activities**.
2. From **Add an optional field**, select **On Success**.
3. From the newly added **On Success**, select **Create new: Action**.
4. From **Add an optional field**, select **On Fail**.
5. From the newly added **On Fail**, select **Create new: Action**.
6. From **Add an optional field**, select **On Late Action**.
7. From the newly added **On Late Action**, select **Create new: Action**.
8. In the right pane, click **Others**.
9. For `DefaultAction1`, do the following:
    a. Change the name to `SuccessSnsAlarm`.
    b. From **Type**, select `SnsAlarm`.
    c. In **Topic Arn**, enter the ARN of the topic that you created (see ARN resource names for Amazon SNS.
    d. Enter a subject and a message.
10. For `DefaultAction2`, do the following:
    a. Change the name to `FailureSnsAlarm`.
    b. From **Type**, select `SnsAlarm`.
    c. In **Topic Arn**, enter the ARN of the topic that you created (see ARN resource names for Amazon SNS.
    d. Enter a subject and a message.
11. For `DefaultAction3`, do the following:

a. Change the name to `LateSnsAlarm`.
b. From **Type**, select `SnsAlarm`.
c. In **Topic Arn**, enter the ARN of the topic that you created (see ARN resource names for Amazon SNS.
d. Enter a subject and a message.

Step 2: Save and Validate Your Pipeline

You can save your pipeline definition at any point during the creation process. As soon as you save your pipeline definition, AWS Data Pipeline looks for syntax errors and missing values in your pipeline definition. If your pipeline is incomplete or incorrect, AWS Data Pipeline generates validation errors and warnings. Warning messages are informational only, but you must fix any error messages before you can activate your pipeline.

To save and validate your pipeline

1. Choose **Save pipeline**.
2. AWS Data Pipeline validates your pipeline definition and returns either success or error or warning messages. If you get an error message, choose **Close** and then, in the right pane, choose **Errors/Warnings**.
3. The **Errors/Warnings** pane lists the objects that failed validation. Choose the plus (**+**) sign next to the object names and look for an error message in red.
4. When you see an error message, go to the specific object pane where you see the error and fix it. For example, if you see an error message in the **DataNodes** object, go to the **DataNodes** pane to fix the error.
5. After you fix the errors listed in the **Errors/Warnings** pane, choose **Save Pipeline**.
6. Repeat the process until your pipeline validates successfully.

Step 3: Activate Your Pipeline

Activate your pipeline to start creating and processing runs. The pipeline starts based on the schedule and period in your pipeline definition.

Important
If activation succeeds, your pipeline is running and might incur usage charges. For more information, see AWS Data Pipeline pricing. To stop incurring usage charges for AWS Data Pipeline, delete your pipeline.

To activate your pipeline

1. Choose **Activate**.
2. In the confirmation dialog box, choose **Close**.

Step 4: Monitor the Pipeline Runs

After you activate your pipeline, you are taken to the **Execution details** page where you can monitor the progress of your pipeline.

To monitor the progress of your pipeline runs

1. Choose **Update** or press F5 to update the status displayed.

Tip
If there are no runs listed, ensure that **Start (in UTC)** and **End (in UTC)** cover the scheduled start and end of your pipeline, and then choose **Update**.

2. When the status of every object in your pipeline is `FINISHED`, your pipeline has successfully completed the scheduled tasks. If you created an SNS notification, you should receive email about the successful completion of this task.
3. If your pipeline doesn't complete successfully, check your pipeline settings for issues. For more information about troubleshooting failed or incomplete instance runs of your pipeline, see Resolving Common Problems (p. 273).

Step 5: Verify the Data Export File

Next, verify that the data export occurred successfully using viewing the output file contents.

To view the export file contents

1. Open the Amazon S3 console.
2. On the **Buckets** pane, click the Amazon S3 bucket that contains your file output (the example pipeline uses the output path **`s3://mybucket/output/MyTable`**) and open the output file with your preferred text editor. The output file name is an identifier value with no extension, such as this example: `ae10f955-fb2f-4790-9b11-fbfea01a871e_000000`.
3. Using your preferred text editor, view the contents of the output file and ensure that there is a data file that corresponds to the DynamoDB source table, such as Id, Price, and ProductCategory. The presence of this text file indicates that the export operation from DynamoDB to the output file occurred successfully.

Step 6: Delete Your Pipeline (Optional)

To stop incurring charges, delete your pipeline. Deleting your pipeline deletes the pipeline definition and all associated objects.

To delete your pipeline

1. On the **List Pipelines** page, select your pipeline.
2. Click **Actions**, and then choose **Delete**.
3. When prompted for confirmation, choose **Delete**.

Copy CSV Data Between Amazon S3 Buckets Using AWS Data Pipeline

After you read What is AWS Data Pipeline? (p. 1) and decide you want to use AWS Data Pipeline to automate the movement and transformation of your data, it is time to get started with creating data pipelines. To help you make sense of how AWS Data Pipeline works, let's walk through a simple task.

This tutorial walks you through the process of creating a data pipeline to copy data from one Amazon S3 bucket to another and then send an Amazon SNS notification after the copy activity completes successfully. You use an EC2 instance managed by AWS Data Pipeline for this copy activity.

Pipeline Objects

The pipeline uses the following objects:

CopyActivity (p. 152)

The activity that AWS Data Pipeline performs for this pipeline (copy CSV data from one Amazon S3 bucket to another).

Important
There are limitations when using the CSV file format with `CopyActivity` and `S3DataNode`. For more information, see CopyActivity (p. 152).

Schedule (p. 252)

The start date, time, and the recurrence for this activity. You can optionally specify the end date and time.

Ec2Resource (p. 204)

The resource (an EC2 instance) that AWS Data Pipeline uses to perform this activity.

S3DataNode (p. 143)

The input and output nodes (Amazon S3 buckets) for this pipeline.

SnsAlarm (p. 250)

The action AWS Data Pipeline must take when the specified conditions are met (send Amazon SNS notifications to a topic after the task finishes successfully).

Contents

Before You Begin

Be sure you've completed the following steps.

- Complete the tasks in Setting Up for AWS Data Pipeline (p. 10).
- (Optional) Set up a VPC for the instance and a security group for the VPC. For more information, see Launching Resources for Your Pipeline into a VPC (p. 48).
- Create an Amazon S3 bucket as a data source.

  For more information, see Create a Bucket in the *Amazon Simple Storage Service Getting Started Guide*.
- Upload your data to your Amazon S3 bucket.

  For more information, see Add an Object to a Bucket in the *Amazon Simple Storage Service Getting Started Guide*.
- Create another Amazon S3 bucket as a data target
- Create a topic for sending email notification and make a note of the topic Amazon Resource Name (ARN). For more information, see Create a Topic in the *Amazon Simple Notification Service Getting Started Guide*.
- (Optional) This tutorial uses the default IAM role policies created by AWS Data Pipeline. If you would rather create and configure your own IAM role policy and trust relationships, follow the instructions described in IAM Roles for AWS Data Pipeline (p. 66).

Copy CSV Data Using the AWS Data Pipeline Console

You can create and use pipelines to copy data from one Amazon S3 bucket to another.

Tasks

Create the Pipeline

First, create the pipeline.

To create the pipeline

1. Open the AWS Data Pipeline console at https://console.aws.amazon.com/datapipeline/.
2. The first screen that you see depends on whether you've created a pipeline in the current region.
   a. If you haven't created a pipeline in this region, the console displays an introductory screen. Choose **Get started now**.
   b. If you've already created a pipeline in this region, the console displays a page that lists your pipelines for the region. Choose **Create new pipeline**.
3. In **Name**, enter a name for your pipeline.
4. (Optional) In **Description**, enter a description for your pipeline.
5. For **Source**, select **Build using Architect**.
6. Under **Schedule**, choose **on pipeline activation**.
7. Under **Pipeline Configuration**, leave logging enabled. Choose the folder icon under **S3 location for logs**, select one of your buckets or folders, and then choose **Select**.

   If you prefer, you can disable logging instead.
8. Under **Security/Access**, leave **IAM roles** set to **Default**.
9. Click **Edit in Architect**.

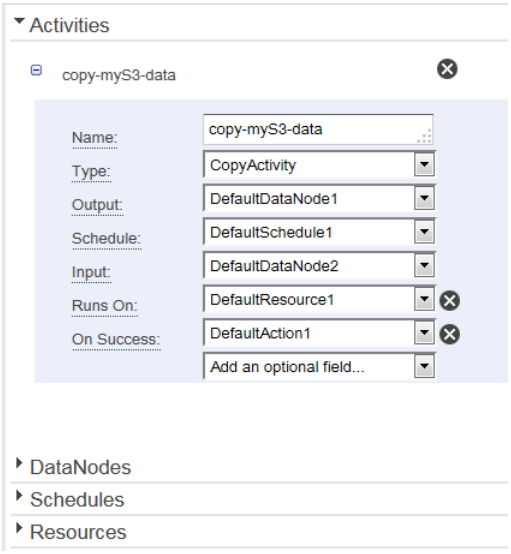
Next, define the `Activity` object in your pipeline definition. When you define the `Activity` object, you also define the objects that AWS Data Pipeline must use to perform this activity.

To configure the activity for your pipeline

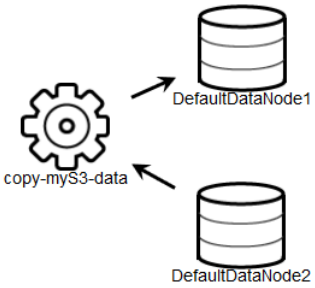
1. Click **Add activity**.
2. In the **Activities** pane:
   a. In the **Name** field, enter a name for the activity, for example, `copy-myS3-data`.
   b. From **Type**, select **CopyActivity**.
   c. From **Output**, select **Create new: DataNode**.
   d. From **Schedule**, select **Create new: Schedule**.
   e. From **Input**, select **Create new: DataNode**.
   f. From **Add an optional field**, select **Runs On**.
   g. From the newly added **Runs On**, select **Create new: Resource**.

h. From **Add an optional field**, select **On Success**.

i. From the newly added **On Success**, select **Create new: Action**.



3. In the left pane, separate the icons by dragging them apart. This is a graphical representation of your pipeline. The arrows indicate the connections between the objects.



Next, configure the input and the output data nodes for your pipeline.

To configure the input and output data nodes for your pipeline

1. In the right pane, click **DataNodes**.
2. For `DefaultDataNode1`, which represents your data source, do the following:
   a. Enter a name for your input node (for example, `MyS3Input`).
   b. From **Type**, select **S3DataNode**.
   c. From **Schedule**, select your schedule (for example, `copy-S3data-schedule`).
   d. From **Add an optional field**, select **File Path**.
   e. In the **File Path** field, enter the path in Amazon S3 for your data source.
3. For `DefaultDataNode2`, which represents your data target, do the following:
   a. Enter a name for your output node (for example, `MyS3Output`).
   b. From **Type**, select **S3DataNode**.
   c. From **Schedule**, select your schedule (for example, `copy-S3data-schedule`).

d. From **Add an optional field**, select **File Path**.
e. In the **File Path** field, enter the path in Amazon S3 for your data target.

Next, configure the resource AWS Data Pipeline must use to perform the copy activity.

To configure the resource

1. In the right pane, click **Resources**.
2. Enter a name for your resource (for example, `CopyDataInstance`).
3. From **Type**, select **Ec2Resource**.
4. From **Schedule**, select your schedule (for example, `copy-S3data-schedule`).
5. Leave **Resource Role** and **Role** set to their default values.

   If you have created your own IAM roles, you can select them if you prefer.
6. [EC2-VPC] From **Add an optional field**, select **Subnet Id**.
7. [EC2-VPC] In **Subnet Id**, enter the ID of the subnet for the EC2 instance.

Next, configure the Amazon SNS notification action that AWS Data Pipeline performs after the copy activity finishes successfully.

To configure the notification action

1. In the right pane, click **Others**.
2. Under `DefaultAction1`, do the following:

   a. Enter a name for your notification (for example, `CopyDataNotice`).
   b. From **Type**, select **SnsAlarm**.
   c. In the **Subject** field, enter the subject line for your notification.
   d. In the **Topic Arn** field, enter the ARN of your topic.
   e. In the **Message** field, enter the message content.
   f. Leave **Role** field set to the default value.

Save and Validate Your Pipeline

You can save your pipeline definition at any point during the creation process. As soon as you save your pipeline definition, AWS Data Pipeline looks for syntax errors and missing values in your pipeline definition. If your pipeline is incomplete or incorrect, AWS Data Pipeline generates validation errors and warnings. Warning messages are informational only, but you must fix any error messages before you can activate your pipeline.

To save and validate your pipeline

1. Choose **Save pipeline**.
2. AWS Data Pipeline validates your pipeline definition and returns either success or error or warning messages. If you get an error message, choose **Close** and then, in the right pane, choose **Errors/Warnings**.
3. The **Errors/Warnings** pane lists the objects that failed validation. Choose the plus (**+**) sign next to the object names and look for an error message in red.
4. When you see an error message, go to the specific object pane where you see the error and fix it. For example, if you see an error message in the **DataNodes** object, go to the **DataNodes** pane to fix the error.

5. After you fix the errors listed in the **Errors/Warnings** pane, choose **Save Pipeline**.
6. Repeat the process until your pipeline validates successfully.

Activate Your Pipeline

Activate your pipeline to start creating and processing runs. The pipeline starts based on the schedule and period in your pipeline definition.

> **Important**
> If activation succeeds, your pipeline is running and might incur usage charges. For more information, see AWS Data Pipeline pricing. To stop incurring usage charges for AWS Data Pipeline, delete your pipeline.

To activate your pipeline

1. Choose **Activate**.
2. In the confirmation dialog box, choose **Close**.

Monitor the Pipeline Runs

After you activate your pipeline, you are taken to the **Execution details** page where you can monitor the progress of your pipeline.

To monitor the progress of your pipeline runs

1. Choose **Update** or press F5 to update the status displayed.

   > **Tip**
   > If there are no runs listed, ensure that **Start (in UTC)** and **End (in UTC)** cover the scheduled start and end of your pipeline, and then choose **Update**.
2. When the status of every object in your pipeline is `FINISHED`, your pipeline has successfully completed the scheduled tasks. If you created an SNS notification, you should receive email about the successful completion of this task.
3. If your pipeline doesn't complete successfully, check your pipeline settings for issues. For more information about troubleshooting failed or incomplete instance runs of your pipeline, see Resolving Common Problems (p. 273).

(Optional) Delete Your Pipeline

To stop incurring charges, delete your pipeline. Deleting your pipeline deletes the pipeline definition and all associated objects.

To delete your pipeline

1. On the **List Pipelines** page, select your pipeline.
2. Click **Actions**, and then choose **Delete**.
3. When prompted for confirmation, choose **Delete**.

Copy CSV Data Using the Command Line

You can create and use pipelines to copy data from one Amazon S3 bucket to another.

Prerequisites

Before you begin, you must complete the following steps:

1. Install and configure a command line interface (CLI). For more information, see Accessing AWS Data Pipeline (p. 2).
2. Ensure that the IAM roles named **DataPipelineDefaultRole** and **DataPipelineDefaultResourceRole** exist. The AWS Data Pipeline console creates these roles for you automatically. If you haven't used the AWS Data Pipeline console at least once, then you must create these roles manually. For more information, see IAM Roles for AWS Data Pipeline (p. 66).

Tasks

- Define a Pipeline in JSON Format (p. 92)
- Upload and Activate the Pipeline Definition (p. 95)

Define a Pipeline in JSON Format

This example scenario shows how to use JSON pipeline definitions and the AWS Data Pipeline CLI to schedule copying data between two Amazon S3 buckets at a specific time interval. This is the full pipeline definition JSON file followed by an explanation for each of its sections.

> **Note**
> We recommend that you use a text editor that can help you verify the syntax of JSON-formatted files, and name the file using the .json file extension.

In this example, for clarity, we skip the optional fields and show only required fields. The complete pipeline JSON file for this example is:

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

Schedule

The pipeline defines a schedule with a begin and end date, along with a period to determine how frequently the activity in this pipeline runs.

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

Amazon S3 Data Nodes

Next, the input S3DataNode pipeline component defines a location for the input files; in this case, an Amazon S3 bucket location. The input S3DataNode component is defined by the following fields:

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

Id

The user-defined name for the input location (a label for your reference only).

Type

The pipeline component type, which is "S3DataNode" to match the location where the data resides, in an Amazon S3 bucket.

Schedule

A reference to the schedule component that we created in the preceding lines of the JSON file labeled “MySchedule”.

Path

The path to the data associated with the data node. The syntax for a data node is determined by its type. For example, the syntax for an Amazon S3 path follows a different syntax that is appropriate for a database table.

Next, the output S3DataNode component defines the output destination location for the data. It follows the same format as the input S3DataNode component, except the name of the component and a different path to indicate the target file.

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

Resource

This is a definition of the computational resource that performs the copy operation. In this example, AWS Data Pipeline should automatically create an EC2 instance to perform the copy task and terminate the resource after the task completes. The fields defined here control the creation and function of the EC2 instance that does the work. The EC2Resource is defined by the following fields:

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

Id

The user-defined name for the pipeline schedule, which is a label for your reference only.

Type

The type of computational resource to perform work; in this case, an EC2 instance. There are other resource types available, such as an EmrCluster type.

Schedule

The schedule on which to create this computational resource.

instanceType

The size of the EC2 instance to create. Ensure that you set the appropriate size of EC2 instance that best matches the load of the work that you want to perform with AWS Data Pipeline. In this case, we set an m1.medium EC2 instance. For more information about the different instance types and when to use each one, see Amazon EC2 Instance Types topic at http://aws.amazon.com/ec2/instance-types/.

Role

The IAM role of the account that accesses resources, such as accessing an Amazon S3 bucket to retrieve data.

resourceRole

The IAM role of the account that creates resources, such as creating and configuring an EC2 instance on your behalf. Role and ResourceRole can be the same role, but separately provide greater granularity in your security configuration.

Activity

The last section in the JSON file is the definition of the activity that represents the work to perform. This example uses `CopyActivity` to copy data from a CSV file in an http://aws.amazon.com/ec2/instance-types/ bucket to another. The `CopyActivity` component is defined by the following fields:

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

Id

The user-defined name for the activity, which is a label for your reference only.

Type

The type of activity to perform, such as MyCopyActivity.

runsOn

The computational resource that performs the work that this activity defines. In this example, we provide a reference to the EC2 instance defined previously. Using the `runsOn` field causes AWS Data Pipeline to create the EC2 instance for you. The `runsOn` field indicates that the resource exists in the AWS infrastructure, while the `workerGroup` value indicates that you want to use your own on-premises resources to perform the work.

Input

The location of the data to copy.

Output

The target location data.

Schedule

The schedule on which to run this activity.

Upload and Activate the Pipeline Definition

You must upload your pipeline definition and activate your pipeline. In the following example commands, replace *`pipeline_name`* with a label for your pipeline and *`pipeline_file`* with the fully-qualified path for the pipeline definition `.json` file.

AWS CLI

To create your pipeline definition and activate your pipeline, use the following create-pipeline command. Note the ID of your pipeline, because you'll use this value with most CLI commands.

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

To upload your pipeline definition, use the following put-pipeline-definition command.

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

If you pipeline validates successfully, the `validationErrors` field is empty. You should review any warnings.

To activate your pipeline, use the following activate-pipeline command.

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

You can verify that your pipeline appears in the pipeline list using the following list-pipelines command.

```
aws datapipeline list-pipelines
```

AWS Data Pipeline CLI

To upload your pipeline definition and activate your pipeline in a single step, use the following command.

```
datapipeline --create pipeline_name --put pipeline_file --activate --force
```

If your pipeline validates successfully, the command displays the following message. Note the ID of your pipeline, because you'll use this value with most AWS Data Pipeline CLI commands.

```
Pipeline with name pipeline_name and id pipeline_id created.
Pipeline definition pipeline_file uploaded.
Pipeline activated.
```

If the command fails, you'll see an error message. For information, see Troubleshooting (p. 270).

You can verify that your pipeline appears in the pipeline list using the following command.

```
datapipeline --list-pipelines
```

Export MySQL Data to Amazon S3 Using AWS Data Pipeline

This tutorial walks you through the process of creating a data pipeline to copy data (rows) from a table in MySQL database to a CSV (comma-separated values) file in an Amazon S3 bucket and then sending an Amazon SNS notification after the copy activity completes successfully. You will use an EC2 instance provided by AWS Data Pipeline for this copy activity.

Pipeline Objects

The pipeline uses the following objects:

- CopyActivity (p. 152)
- Ec2Resource (p. 204)
- MySqlDataNode (p. 135)
- S3DataNode (p. 143)
- SnsAlarm (p. 250)

Contents

Before You Begin

Be sure you've completed the following steps.

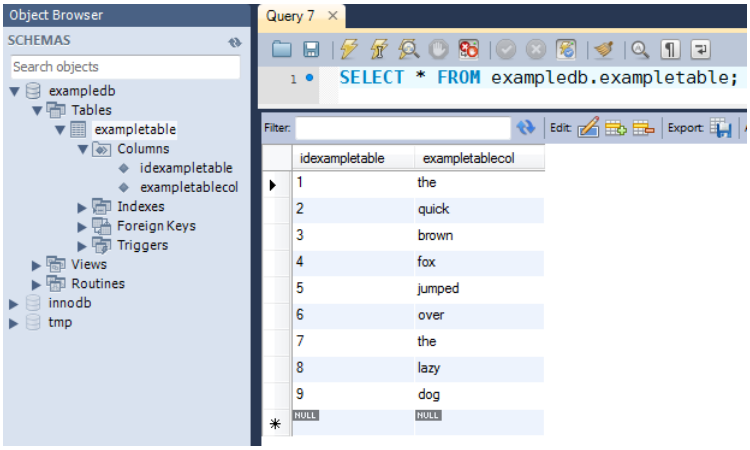
- Complete the tasks in Setting Up for AWS Data Pipeline (p. 10).
- (Optional) Set up a VPC for the instance and a security group for the VPC. For more information, see Launching Resources for Your Pipeline into a VPC (p. 48).
- Create an Amazon S3 bucket as a data output.

  For more information, see Create a Bucket in *Amazon Simple Storage Service Getting Started Guide*.
- Create and launch a MySQL database instance as your data source.

  For more information, see Launch a DB Instance in the *Amazon Relational Database Service Getting Started Guide*. After you have an Amazon RDS instance, see Create a Table in the MySQL documentation.

  Note
  Make a note of the user name and the password you used for creating the MySQL instance. After you've launched your MySQL database instance, make a note of the instance's endpoint. You'll need this information later.
- Connect to your MySQL database instance, create a table, and then add test data values to the newly created table.

  For illustration purposes, we created this tutorial using a MySQL table with the following configuration and sample data. The following screen shot is from MySQL Workbench 5.2 CE:



For more information, see Create a Table in the MySQL documentation and the MySQL Workbench product page.

- Create a topic for sending email notification and make a note of the topic Amazon Resource Name (ARN). For more information, see Create a Topic in *Amazon Simple Notification Service Getting Started Guide*.
- (Optional) This tutorial uses the default IAM role policies created by AWS Data Pipeline. If you would rather create and configure your IAM role policy and trust relationships, follow the instructions described in IAM Roles for AWS Data Pipeline (p. 66).

Copy MySQL Data Using the AWS Data Pipeline Console

You can create a pipeline to copy data from a MySQL table to a file in an Amazon S3 bucket.

Tasks

Create the Pipeline

First, create the pipeline. The pipeline must be created in the same region as your target RDS instance.

To create your pipeline

1. Open the AWS Data Pipeline console at https://console.aws.amazon.com/datapipeline/.
2. The first screen that you see depends on whether you've created a pipeline in the current region.
   a. If you haven't created a pipeline in this region, the console displays an introductory screen. Choose **Get started now**.
   b. If you've already created a pipeline in this region, the console displays a page that lists your pipelines for the region. Choose **Create new pipeline**.
3. In **Name**, enter a name for your pipeline.
4. (Optional) In **Description**, enter a description for your pipeline.
5. For **Source**, select **Build using a template**, and then select the following template: **Full copy of RDS MySQL table to S3**.
6. Under the **Parameters** section, which opened when you selected the template, do the following:
   a. For **DBInstance ID**, enter the DB instance name of the Aurora DB instance you want to use to copy data from the Aurora cluster.

      To locate the endpoint details for your DB instance, see Connecting to a DB Instance Running the MySQL Database Engine in the *Amazon Relational Database Service User Guide*.
   b. For **RDS MySQL username**, enter the user name you used when you created your MySQL database instance.

   c. In the **RDS MySQL password** field, enter the password you used when you created your DB instance.
   d. In the **EC2 instance type** field, enter the instance type for your EC2 instance.
   e. Click the folder icon next to **Output S3 folder**, select one of your buckets or folders, and then click **Select**.
7. Under **Schedule**, choose **on pipeline activation**.
8. Under **Pipeline Configuration**, leave logging enabled. Choose the folder icon under **S3 location for logs**, select one of your buckets or folders, and then choose **Select**.

   If you prefer, you can disable logging instead.
9. Under **Security/Access**, leave **IAM roles** set to **Default**.
10. Click **Edit in Architect**.
11. In the left pane, separate the icons by dragging them apart. This is a graphical representation of your pipeline. The arrows indicate the connections between the objects.

Next, configure the database name setting, which currently is not present on the available template.

1. In the left pane, click **RDSDatabase**.
2. In the right pane, under the rds_mysql section, for **Add an optional field...** choose **Database Name**.
3. Type the **Database Name** of your target database and add optional fields.

You can configure the Amazon SNS notification action AWS Data Pipeline performs after the copy activity finishes successfully.

To configure the Amazon SNS notification action

1. In the right pane, click **Activities**.
2. From **Add an optional field**, select **On Success**.
3. From the newly added **On Success**, select **Create new: Action**.
4. In the right pane, click **Others**.
5. Under `DefaultAction1`, do the following:

   a. Enter a name for your notification (for example, `CopyDataNotice`).
   b. From **Type**, select **SnsAlarm**.
   c. In the **Message** field, enter the message content.
   d. In the **Subject** field, enter the subject line for your notification.
   e. In the **Topic Arn** field, enter the ARN of your topic.
   f. Leave **Role** field set to the default value.

Save and Validate Your Pipeline

You can save your pipeline definition at any point during the creation process. As soon as you save your pipeline definition, AWS Data Pipeline looks for syntax errors and missing values in your pipeline definition. If your pipeline is incomplete or incorrect, AWS Data Pipeline generates validation errors and warnings.

Warning messages are informational only, but you must fix any error messages before you can activate your pipeline.

To save and validate your pipeline

1. Choose **Save pipeline**.
2. AWS Data Pipeline validates your pipeline definition and returns either success or error or warning messages. If you get an error message, choose **Close** and then, in the right pane, choose **Errors/Warnings**.
3. The **Errors/Warnings** pane lists the objects that failed validation. Choose the plus (**+**) sign next to the object names and look for an error message in red.
4. When you see an error message, go to the specific object pane where you see the error and fix it. For example, if you see an error message in the **DataNodes** object, go to the **DataNodes** pane to fix the error.
5. After you fix the errors listed in the **Errors/Warnings** pane, choose **Save Pipeline**.
6. Repeat the process until your pipeline validates successfully.

Verify Your Pipeline Definition

It is important that you verify that your pipeline was correctly initialized from your definitions before you activate it.

To verify your pipeline definition

1. On the **List Pipelines** page, look for your newly-created pipeline.

   AWS Data Pipeline has created a unique **Pipeline ID** for your pipeline definition.

   The **Schedule State** column in the row listing your pipeline should show `PENDING`.
2. Choose the triangle icon next to your pipeline. A pipeline summary pane below shows the details of your pipeline runs. Because your pipeline is not yet activated, you are not likely to see any execution details. However, you will see the configuration of the pipeline definition.

Activate Your Pipeline

Activate your pipeline to start creating and processing runs. The pipeline starts based on the schedule and period in your pipeline definition.

> **Important**
> If activation succeeds, your pipeline is running and might incur usage charges. For more information, see AWS Data Pipeline pricing. To stop incurring usage charges for AWS Data Pipeline, delete your pipeline.

To activate your pipeline

1. Choose **Activate**.
2. In the confirmation dialog box, choose **Close**.

Monitor the Pipeline Runs

After you activate your pipeline, you are taken to the **Execution details** page where you can monitor the progress of your pipeline.

To monitor the progress of your pipeline runs

1. Choose **Update** or press F5 to update the status displayed.

   Tip
   If there are no runs listed, ensure that **Start (in UTC)** and **End (in UTC)** cover the scheduled start and end of your pipeline, and then choose **Update**.
2. When the status of every object in your pipeline is `FINISHED`, your pipeline has successfully completed the scheduled tasks. If you created an SNS notification, you should receive email about the successful completion of this task.
3. If your pipeline doesn't complete successfully, check your pipeline settings for issues. For more information about troubleshooting failed or incomplete instance runs of your pipeline, see Resolving Common Problems (p. 273).

(Optional) Delete Your Pipeline

To stop incurring charges, delete your pipeline. Deleting your pipeline deletes the pipeline definition and all associated objects.

To delete your pipeline

1. On the **List Pipelines** page, select your pipeline.
2. Click **Actions**, and then choose **Delete**.
3. When prompted for confirmation, choose **Delete**.

Copy MySQL Data Using the Command Line

You can create a pipeline to copy data from a MySQL table to a file in an Amazon S3 bucket.

Prerequisites

Before you begin, you must complete the following steps:

1. Install and configure a command line interface (CLI). For more information, see Accessing AWS Data Pipeline (p. 2).
2. Ensure that the IAM roles named **DataPipelineDefaultRole** and **DataPipelineDefaultResourceRole** exist. The AWS Data Pipeline console creates these roles for you automatically. If you haven't used the AWS Data Pipeline console at least once, then you must create these roles manually. For more information, see IAM Roles for AWS Data Pipeline (p. 66).
3. Set up an Amazon S3 bucket and an Amazon RDS instance. For more information, see Before You Begin (p. 97).

Tasks

- Define a Pipeline in JSON Format (p. 101)
- Upload and Activate the Pipeline Definition (p. 106)

Define a Pipeline in JSON Format

This example scenario shows how to use JSON pipeline definitions and the AWS Data Pipeline CLI to copy data (rows) from a table in a MySQL database to a CSV (comma-separated values) file in an Amazon S3 bucket at a specified time interval.

This is the full pipeline definition JSON file followed by an explanation for each of its sections.

Note
We recommend that you use a text editor that can help you verify the syntax of JSON-formatted files, and name the file using the .json file extension.

```
{
  "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 Data Node

The input MySqlDataNode pipeline component defines a location for the input data; in this case, an Amazon RDS instance. The input MySqlDataNode component is defined by the following fields:

```
{
  "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

The user-defined name, which is a label for your reference only.

Username

The user name of the database account that has sufficient permission to retrieve data from the database table. Replace *my-username* with the name of your user account.

Schedule

A reference to the schedule component that we created in the preceding lines of the JSON file.

Name

The user-defined name, which is a label for your reference only.

*Password

The password for the database account with the asterisk prefix to indicate that AWS Data Pipeline must encrypt the password value. Replace *my-password* with the correct password for your user account. The password field is preceded by the asterisk special character. For more information, see Special Characters (p. 128).

Table

The name of the database table that contains the data to copy. Replace *table-name* with the name of your database table.

connectionString

The JDBC connection string for the CopyActivity object to connect to the database.

selectQuery

A valid SQL SELECT query that specifies which data to copy from the database table. Note that `#{table}` is an expression that re-uses the table name provided by the "table" variable in the preceding lines of the JSON file.

Type

The SqlDataNode type, which is an Amazon RDS instance using MySQL in this example.

Note
The MySqlDataNode type is deprecated. While you can still use MySqlDataNode, we recommend using SqlDataNode.

Amazon S3 Data Node

Next, the S3Output pipeline component defines a location for the output file; in this case a CSV file in an Amazon S3 bucket location. The output S3DataNode component is defined by the following fields:

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

Id

The user-defined ID, which is a label for your reference only.

Schedule

A reference to the schedule component that we created in the preceding lines of the JSON file.

filePath

The path to the data associated with the data node, which is an CSV output file in this example.

Name

The user-defined name, which is a label for your reference only.

Type

The pipeline object type, which is S3DataNode to match the location where the data resides, in an Amazon S3 bucket.

Resource

This is a definition of the computational resource that performs the copy operation. In this example, AWS Data Pipeline should automatically create an EC2 instance to perform the copy task and terminate the resource after the task completes. The fields defined here control the creation and function of the EC2 instance that does the work. The EC2Resource is defined by the following fields:

```
{
  "id": "Ec2ResourceId116",
  "schedule": {
    "ref": "ScheduleId113"
  },
  "name": "My EC2 Resource",
  "role": "DataPipelineDefaultRole",
  "type": "Ec2Resource",
  "resourceRole": "DataPipelineDefaultResourceRole"
},
```

Id

The user-defined ID, which is a label for your reference only.

Schedule

The schedule on which to create this computational resource.

Name

The user-defined name, which is a label for your reference only.

Role

The IAM role of the account that accesses resources, such as accessing an Amazon S3 bucket to retrieve data.

Type

The type of computational resource to perform work; in this case, an EC2 instance. There are other resource types available, such as an EmrCluster type.

resourceRole

The IAM role of the account that creates resources, such as creating and configuring an EC2 instance on your behalf. Role and ResourceRole can be the same role, but separately provide greater granularity in your security configuration.

Activity

The last section in the JSON file is the definition of the activity that represents the work to perform. In this case we use a CopyActivity component to copy data from a file in an Amazon S3 bucket to another file. The CopyActivity component is defined by the following fields:

```
{
  "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

The user-defined ID, which is a label for your reference only

Input

The location of the MySQL data to copy

Schedule

The schedule on which to run this activity

Name

The user-defined name, which is a label for your reference only

runsOn

The computational resource that performs the work that this activity defines. In this example, we provide a reference to the EC2 instance defined previously. Using the `runsOn` field causes AWS Data Pipeline to create the EC2 instance for you. The `runsOn` field indicates that the resource exists in the AWS infrastructure, while the workerGroup value indicates that you want to use your own on-premises resources to perform the work.

onSuccess

The SnsAlarm (p. 250) to send if the activity completes successfully

onFail

The SnsAlarm (p. 250) to send if the activity fails

Output

The Amazon S3 location of the CSV output file

Type

The type of activity to perform.

Upload and Activate the Pipeline Definition

You must upload your pipeline definition and activate your pipeline. In the following example commands, replace *`pipeline_name`* with a label for your pipeline and *`pipeline_file`* with the fully-qualified path for the pipeline definition `.json` file.

AWS CLI

To create your pipeline definition and activate your pipeline, use the following create-pipeline command. Note the ID of your pipeline, because you'll use this value with most CLI commands.

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

To upload your pipeline definition, use the following put-pipeline-definition command.

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

If you pipeline validates successfully, the `validationErrors` field is empty. You should review any warnings.

To activate your pipeline, use the following activate-pipeline command.

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

You can verify that your pipeline appears in the pipeline list using the following list-pipelines command.

```
aws datapipeline list-pipelines
```

AWS Data Pipeline CLI

To upload your pipeline definition and activate your pipeline in a single step, use the following command.

```
datapipeline --create pipeline_name --put pipeline_file --activate --force
```

If your pipeline validates successfully, the command displays the following message. Note the ID of your pipeline, because you'll use this value with most AWS Data Pipeline CLI commands.

```
Pipeline with name pipeline_name and id pipeline_id created.
Pipeline definition pipeline_file uploaded.
Pipeline activated.
```

If the command fails, you'll see an error message. For information, see Troubleshooting (p. 270).

You can verify that your pipeline appears in the pipeline list using the following command.

```
datapipeline --list-pipelines
```

Copy Data to Amazon Redshift Using AWS Data Pipeline

This tutorial walks you through the process of creating a pipeline that periodically moves data from Amazon S3 to Amazon Redshift using either the **Copy to Redshift** template in the AWS Data Pipeline console, or a pipeline definition file with the AWS Data Pipeline CLI.

Amazon S3 is a web service that enables you to store data in the cloud. For more information, see the Amazon Simple Storage Service Console User Guide. Amazon Redshift is a data warehouse service in the cloud. For more information, see the Amazon Redshift Cluster Management Guide.

Contents

Before You Begin

This tutorial has several prerequisites. After completing the following steps, you can continue the tutorial using either the console or the CLI.

To set up for the tutorial

1. Complete the tasks in Setting Up for AWS Data Pipeline (p. 10).
2. Create a security group.
   a. Open the Amazon EC2 console.
   b. In the navigation pane, click **Security Groups**.
   c. Click **Create Security Group**.
   d. Specify a name and description for the security group.
   e. [EC2-Classic] Select `No VPC` for **VPC**.
   f. [EC2-VPC] Select the ID of your VPC for **VPC**.
   g. Click **Create**.
3. [EC2-Classic] Create an Amazon Redshift cluster security group and specify the Amazon EC2 security group.
   a. Open the Amazon Redshift console.
   b. In the navigation pane, click **Security Groups**.
   c. Click **Create Cluster Security Group**.
   d. In the **Create Cluster Security Group** dialog box, specify a name and description for the cluster security group.
   e. Click the name of the new cluster security group.
   f. Click **Add Connection Type**.
   g. In the **Add Connection Type** dialog box, select **EC2 Security Group** from **Connection Type**, select the security group that you created from **EC2 Security Group Name**, and then click **Authorize**.
4. [EC2-VPC] Create an Amazon Redshift cluster security group and specify the VPC security group.
   a. Open the Amazon EC2 console.
   b. In the navigation pane, click **Security Groups**.
   c. Click **Create Security Group**.
   d. In the **Create Security Group** dialog box, specify a name and description for the security group, and select the ID of your VPC for **VPC**.
   e. Click **Add Rule**. Specify the type, protocol, and port range, and start typing the ID of the security group in **Source**. Select the security group that you created in the second step.
   f. Click **Create**.
5. Select an existing Amazon Redshift database, or create a new one. The following is a summary of the steps; for more information, see Creating a Cluster in the *Amazon Redshift Cluster Management Guide*.
   a. Open the Amazon Redshift console.

b. Click **Launch Cluster**.
c. Provide the required details for your cluster, and then click **Continue**.
d. Provide the node configuration, and then click **Continue**.
e. On the page for additional configuration information, select the cluster security group that you created, and then click **Continue**.
f. Review the specifications for your cluster, and then click **Launch Cluster**.

Copy Data to Amazon Redshift Using the AWS Data Pipeline Console

You can create a pipeline to copy data from Amazon S3 to Amazon Redshift. You'll create a new table in Amazon Redshift, and then use AWS Data Pipeline to transfer data to this table from a public Amazon S3 bucket, which contains sample input data in CSV format. The logs are saved to an Amazon S3 bucket that you own.

Amazon S3 is a web service that enables you to store data in the cloud. For more information, see the Amazon Simple Storage Service Console User Guide. Amazon Redshift is a data warehouse service in the cloud. For more information, see the Amazon Redshift Cluster Management Guide.

Prerequisites

Before you start this tutorial, complete the prerequisites described in Before You Begin (p. 108).

Tasks

Create the Pipeline

First, create the pipeline.

To create the pipeline

1. Open the AWS Data Pipeline console at https://console.aws.amazon.com/datapipeline/.
2. The first screen that you see depends on whether you've created a pipeline in the current region.
   a. If you haven't created a pipeline in this region, the console displays an introductory screen. Choose **Get started now**.
   b. If you've already created a pipeline in this region, the console displays a page that lists your pipelines for the region. Choose **Create new pipeline**.
3. In **Name**, enter a name for your pipeline.
4. (Optional) In **Description**, enter a description for your pipeline.
5. For **Source**, select **Build using a template**, and then select the following template: **Load data from S3 into Redshift**.
6. Under **Parameters**, provide information about your input folder in Amazon S3 and the Amazon Redshift database that you created.

7. Under **Schedule**, choose **on pipeline activation**.
8. Under **Pipeline Configuration**, leave logging enabled. Choose the folder icon under **S3 location for logs**, select one of your buckets or folders, and then choose **Select**.

   If you prefer, you can disable logging instead.
9. Under **Security/Access**, leave **IAM roles** set to **Default**.
10. Click **Activate**.

    If you prefer, you can choose **Edit in Architect** to modify this pipeline. For example, you can add preconditions.

Save and Validate Your Pipeline

You can save your pipeline definition at any point during the creation process. As soon as you save your pipeline definition, AWS Data Pipeline looks for syntax errors and missing values in your pipeline definition. If your pipeline is incomplete or incorrect, AWS Data Pipeline generates validation errors and warnings. Warning messages are informational only, but you must fix any error messages before you can activate your pipeline.

To save and validate your pipeline

1. Choose **Save pipeline**.
2. AWS Data Pipeline validates your pipeline definition and returns either success or error or warning messages. If you get an error message, choose **Close** and then, in the right pane, choose **Errors/Warnings**.
3. The **Errors/Warnings** pane lists the objects that failed validation. Choose the plus (**+**) sign next to the object names and look for an error message in red.
4. When you see an error message, go to the specific object pane where you see the error and fix it. For example, if you see an error message in the **DataNodes** object, go to the **DataNodes** pane to fix the error.
5. After you fix the errors listed in the **Errors/Warnings** pane, choose **Save Pipeline**.
6. Repeat the process until your pipeline validates successfully.

Activate Your Pipeline

Activate your pipeline to start creating and processing runs. The pipeline starts based on the schedule and period in your pipeline definition.

> **Important**
> If activation succeeds, your pipeline is running and might incur usage charges. For more information, see AWS Data Pipeline pricing. To stop incurring usage charges for AWS Data Pipeline, delete your pipeline.

To activate your pipeline

1. Choose **Activate**.
2. In the confirmation dialog box, choose **Close**.

Monitor the Pipeline Runs

After you activate your pipeline, you are taken to the **Execution details** page where you can monitor the progress of your pipeline.

To monitor the progress of your pipeline runs

1. Choose **Update** or press F5 to update the status displayed.

   Tip
   If there are no runs listed, ensure that **Start (in UTC)** and **End (in UTC)** cover the scheduled start and end of your pipeline, and then choose **Update**.
2. When the status of every object in your pipeline is `FINISHED`, your pipeline has successfully completed the scheduled tasks. If you created an SNS notification, you should receive email about the successful completion of this task.
3. If your pipeline doesn't complete successfully, check your pipeline settings for issues. For more information about troubleshooting failed or incomplete instance runs of your pipeline, see Resolving Common Problems (p. 273).

(Optional) Delete Your Pipeline

To stop incurring charges, delete your pipeline. Deleting your pipeline deletes the pipeline definition and all associated objects.

To delete your pipeline

1. On the **List Pipelines** page, select your pipeline.
2. Click **Actions**, and then choose **Delete**.
3. When prompted for confirmation, choose **Delete**.

Copy Data to Amazon Redshift Using the Command Line

This tutorial demonstrates how to copy data from Amazon S3 to Amazon Redshift. You'll create a new table in Amazon Redshift, and then use AWS Data Pipeline to transfer data to this table from a public Amazon S3 bucket, which contains sample input data in CSV format. The logs are saved to an Amazon S3 bucket that you own.

Amazon S3 is a web service that enables you to store data in the cloud. For more information, see the Amazon Simple Storage Service Console User Guide. Amazon Redshift is a data warehouse service in the cloud. For more information, see the Amazon Redshift Cluster Management Guide.

Prerequisites

Before you begin, you must complete the following steps:

1. Install and configure a command line interface (CLI). For more information, see Accessing AWS Data Pipeline (p. 2).
2. Ensure that the IAM roles named **DataPipelineDefaultRole** and **DataPipelineDefaultResourceRole** exist. The AWS Data Pipeline console creates these roles for you automatically. If you haven't used the AWS Data Pipeline console at least once, then you must create these roles manually. For more information, see IAM Roles for AWS Data Pipeline (p. 66).
3. Set up an Amazon Redshift database. For more information, see Before You Begin (p. 108).

Tasks

Define a Pipeline in JSON Format

This example scenario shows how to copy data from an Amazon S3 bucket to Amazon Redshift.

This is the full pipeline definition JSON file followed by an explanation for each of its sections. We recommend that you use a text editor that can help you verify the syntax of JSON-formatted files, and name the file using the `.json` file extension.

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

For more information about these objects, see the following documentation.

Objects

- Data Nodes (p. 113)
- Resource (p. 115)
- Activity (p. 116)

Data Nodes

This example uses an input data node, an output data node, and a database.

Input Data Node

The input `S3DataNode` pipeline component defines the location of the input data in Amazon S3 and the data format of the input data. For more information, see S3DataNode (p. 143).

This input component is defined by the following fields:

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

```
    "ref": "CSVId1"
  },
  "type": "S3DataNode"
},
```

`id`

The user-defined ID, which is a label for your reference only.

`schedule`

A reference to the schedule component.

`filePath`

The path to the data associated with the data node, which is an CSV input file in this example.

`name`

The user-defined name, which is a label for your reference only.

`dataFormat`

A reference to the format of the data for the activity to process.

Output Data Node

The output `RedshiftDataNode` pipeline component defines a location for the output data; in this case, a table in an Amazon Redshift database. For more information, see RedshiftDataNode (p. 139). This output component is defined by the following fields:

```
{
  "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`

The user-defined ID, which is a label for your reference only.

`schedule`

A reference to the schedule component.

`tableName`

The name of the Amazon Redshift table.

`name`

The user-defined name, which is a label for your reference only.

`createTableSql`

A SQL expression to create the table in the database.

`database`

A reference to the Amazon Redshift database.

Database

The `RedshiftDatabase` component is defined by the following fields. For more information, see RedshiftDatabase (p. 240).

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

`id`

The user-defined ID, which is a label for your reference only.

`databaseName`

The name of the logical database.

`username`

The user name to connect to the database.

`name`

The user-defined name, which is a label for your reference only.

`password`

The password to connect to the database.

`clusterId`

The ID of the Redshift cluster.

Resource

This is a definition of the computational resource that performs the copy operation. In this example, AWS Data Pipeline should automatically create an EC2 instance to perform the copy task and terminate the instance after the task completes. The fields defined here control the creation and function of the instance that does the work. For more information, see Ec2Resource (p. 204).

The `Ec2Resource` is defined by the following fields:

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

```
  "type": "Ec2Resource"
},
```

`id`

The user-defined ID, which is a label for your reference only.

`schedule`

The schedule on which to create this computational resource.

`securityGroups`

The security group to use for the instances in the resource pool.

`name`

The user-defined name, which is a label for your reference only.

`role`

The IAM role of the account that accesses resources, such as accessing an Amazon S3 bucket to retrieve data.

`logUri`

The Amazon S3 destination path to back up Task Runner logs from the `Ec2Resource`.

`resourceRole`

The IAM role of the account that creates resources, such as creating and configuring an EC2 instance on your behalf. Role and ResourceRole can be the same role, but separately provide greater granularity in your security configuration.

Activity

The last section in the JSON file is the definition of the activity that represents the work to perform. In this case, we use a `RedshiftCopyActivity` component to copy data from Amazon S3 to Amazon Redshift. For more information, see RedshiftCopyActivity (p. 187).

The `RedshiftCopyActivity` component is defined by the following fields:

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

`id`

The user-defined ID, which is a label for your reference only.

`input`

A reference to the Amazon S3 source file.

`schedule`

The schedule on which to run this activity.

`insertMode`

The insert type (`KEEP_EXISTING`, `OVERWRITE_EXISTING`, or `TRUNCATE`).

`name`

The user-defined name, which is a label for your reference only.

`runsOn`

The computational resource that performs the work that this activity defines.

`output`

A reference to the Amazon Redshift destination table.

Upload and Activate the Pipeline Definition

You must upload your pipeline definition and activate your pipeline. In the following example commands, replace *`pipeline_name`* with a label for your pipeline and *`pipeline_file`* with the fully-qualified path for the pipeline definition `.json` file.

AWS CLI

To create your pipeline definition and activate your pipeline, use the following create-pipeline command. Note the ID of your pipeline, because you'll use this value with most CLI commands.

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

To upload your pipeline definition, use the following put-pipeline-definition command.

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

If you pipeline validates successfully, the `validationErrors` field is empty. You should review any warnings.

To activate your pipeline, use the following activate-pipeline command.

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

You can verify that your pipeline appears in the pipeline list using the following list-pipelines command.

```
aws datapipeline list-pipelines
```

AWS Data Pipeline CLI

To upload your pipeline definition and activate your pipeline in a single step, use the following command.

```
datapipeline --create pipeline_name --put pipeline_file --activate --force
```

If your pipeline validates successfully, the command displays the following message. Note the ID of your pipeline, because you'll use this value with most AWS Data Pipeline CLI commands.

```
Pipeline with name pipeline_name and id pipeline_id created.
Pipeline definition pipeline_file uploaded.
Pipeline activated.
```

If the command fails, you'll see an error message. For information, see Troubleshooting (p. 270).

You can verify that your pipeline appears in the pipeline list using the following command.

```
datapipeline --list-pipelines
```

Pipeline Expressions and Functions

This section explains the syntax for using expressions and functions in pipelines, including the associated data types.

Simple Data Types

The following types of data can be set as field values.

Types

- DateTime (p. 119)
- Numeric (p. 119)
- Object References (p. 119)
- Period (p. 120)
- String (p. 120)

DateTime

AWS Data Pipeline supports the date and time expressed in "YYYY-MM-DDTHH:MM:SS" format in UTC/GMT only. The following example sets the `startDateTime` field of a `Schedule` object to `1/15/2012, 11:59 p.m.`, in the UTC/GMT timezone.

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

Numeric

AWS Data Pipeline supports both integers and floating-point values.

Object References

An object in the pipeline definition. This can either be the current object, the name of an object defined elsewhere in the pipeline, or an object that lists the current object in a field, referenced by the `node` keyword. For more information about `node`, see Referencing Fields and Objects (p. 120). For more information about the pipeline object types, see Pipeline Object Reference (p. 130).

Period

Indicates how often a scheduled event should run. It's expressed in the format "*N* [`years`|`months`|`weeks`|`days`|`hours`|`minutes`]", where *N* is a positive integer value.

The minimum period is 15 minutes and the maximum period is 3 years.

The following example sets the `period` field of the `Schedule` object to 3 hours. This creates a schedule that runs every three hours.

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

String

Standard string values. Strings must be surrounded by double quotes ("). You can use the backslash character (\) to escape characters in a string. Multiline strings are not supported.

The following examples show examples of valid string values for the `id` field.

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

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

Strings can also contain expressions that evaluate to string values. These are inserted into the string, and are delimited with: "#{" and "}". The following example uses an expression to insert the name of the current object into a path.

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

For more information about using expressions, see Referencing Fields and Objects (p. 120) and Expression Evaluation (p. 123).

Expressions

Expressions enable you to share a value across related objects. Expressions are processed by the AWS Data Pipeline web service at runtime, ensuring that all expressions are substituted with the value of the expression.

Expressions are delimited by: "#{" and "}". You can use an expression in any pipeline definition object where a string is legal. If a slot is a reference or one of type ID, NAME, TYPE, SPHERE, its value is not evaluated and used verbatim.

The following expression calls one of the AWS Data Pipeline functions. For more information, see Expression Evaluation (p. 123).

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

Referencing Fields and Objects

Expressions can use fields of the current object where the expression exists, or fields of another object that is linked by a reference.

In the following example, the `filePath` field references the `id` field in the same object to form a file name. The value of `filePath` evaluates to "s3://mybucket/ExampleDataNode.csv".

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

To use a field that exists on another object linked by a reference, use the `node` keyword. This keyword is only available with alarm and precondition objects.

Continuing with the previous example, an expression in an `SnsAlarm` can refer to the date and time range in a `Schedule`, because the `S3DataNode` references both. Specifically, `FailureNotify`'s `message` field can use the `@scheduledStartTime` and `@scheduledEndTime` runtime fields from `ExampleSchedule`, because `ExampleDataNode`'s `onFail` field references `FailureNotify` and its `schedule` field references `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
You can create pipelines that have dependencies, such as tasks in your pipeline that depend on the work of other systems or tasks. If your pipeline requires certain resources, add those dependencies to the pipeline using preconditions that you associate with data nodes and tasks so your pipelines are easier to debug and more resilient. Additionally, keep your dependencies within a single pipeline when possible, because cross-pipeline troubleshooting is difficult.

Nested Expressions

AWS Data Pipeline allows you to nest values to create more complex expressions. For example, to perform a time calculation (subtract 30 minutes from the `scheduledStartTime`) and format the result to use in a pipeline definition, you could use the following expression in an activity:

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

and using the `node` prefix if the expression is part of an SnsAlarm or Precondition:

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

Lists

Expressions can be evaluated on lists and functions on lists. For example, assume that a list is defined like the following: `"myList":["one","two"]`. If this list is used in the expression `#{'this is ' + myList}`, it will evaluate to `["this is one", "this is two"]`. If you have two lists, Data Pipeline will ultimately flatten them in their evaluation. For example, if `myList1` is defined as `[1,2]` and `myList2` is defined as `[3,4]` then the expression `[#{myList1}, #{myList2}]` will evaluate to `[1,2,3,4]`.

Node Expression

AWS Data Pipeline uses the `#{node.*}` expression in either `SnsAlarm` or `PreCondition` for a back-reference to a pipeline component's parent object. Since `SnsAlarm` and `PreCondition` are referenced from an activity or resource with no reference back from them, `node` provides the way to refer to the referrer. For example, the following pipeline definition demonstrates how a failure notification can use `node` to make a reference to its parent, in this case `ShellCommandActivity`, and include the parent's scheduled start and end times in the `SnsAlarm` message. The scheduledStartTime reference on ShellCommandActivity does not require the `node` prefix because scheduledStartTime refers to itself.

Note
The fields preceded by the AT (@) sign indicate those fields are runtime fields.

```
{
  "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 supports transitive references for user-defined fields, but not runtime fields. A transitive reference is a reference between two pipeline components that depends on another pipeline component as the intermediary. The following example shows a reference to a transitive user-defined field and a reference to a non-transitive runtime field, both of which are valid. For more information, see User-Defined Fields (p. 56).

```
{
  "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"
}
```

Expression Evaluation

AWS Data Pipeline provides a set of functions that you can use to calculate the value of a field. The following example uses the `makeDate` function to set the `startDateTime` field of a `Schedule` object to `"2011-05-24T0:00:00"` GMT/UTC.

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

Mathematical Functions

The following functions are available for working with numerical values.

Function	Description
+	Addition. Example: `#{1 + 2}` Result: `3`
-	Subtraction. Example: `#{1 - 2}` Result: `-1`
*	Multiplication. Example: `#{1 * 2}` Result: `2`
/	Division. If you divide two integers, the result is truncated. Example: `#{1 / 2}`, Result: `0` Example: `#{1.0 / 2}`, Result: `.5`
^	Exponent. Example: `#{2 ^ 2}` Result: `4.0`

String Functions

The following functions are available for working with string values.

Function	Description
+	Concatenation. Non-string values are first converted to strings. Example: `#{"hel" + "lo"}`

Function	Description
	Result: `"hello"`

Date and Time Functions

The following functions are available for working with DateTime values. For the examples, the value of `myDateTime` is `May 24, 2011 @ 5:10 pm GMT`.

Note
The date/time format for AWS Data Pipeline is Joda Time, which is a replacement for the Java date and time classes. For more information, see Joda Time - Class DateTimeFormat.

Function	Description
`int day(DateTime myDateTime)`	Gets the day of the DateTime value as an integer. Example: `#{day(myDateTime)}` Result: `24`
`int dayOfYear(DateTime myDateTime)`	Gets the day of the year of the DateTime value as an integer. Example: `#{dayOfYear(myDateTime)}` Result: `144`
`DateTime firstOfMonth(DateTime myDateTime)`	Creates a DateTime object for the start of the month in the specified DateTime. Example: `#{firstOfMonth(myDateTime)}` Result: `"2011-05-01T17:10:00z"`
`String format(DateTime myDateTime,String format)`	Creates a String object that is the result of converting the specified DateTime using the specified format string. Example: `#{format(myDateTime,'YYYY-MM-dd HH:mm:ss z')}` Result: `"2011-05-24T17:10:00 UTC"`
`int hour(DateTime myDateTime)`	Gets the hour of the DateTime value as an integer. Example: `#{hour(myDateTime)}` Result: `17`

Function	Description
`DateTime makeDate(int year,int month,int day)`	Creates a DateTime object, in UTC, with the specified year, month, and day, at midnight. Example: `#{makeDate(2011,5,24)}` Result: `"2011-05-24T0:00:00z"`
`DateTime makeDateTime(int year,int month,int day,int hour,int minute)`	Creates a DateTime object, in UTC, with the specified year, month, day, hour, and minute. Example: `#{makeDateTime(2011,5,24,14,21)}` Result: `"2011-05-24T14:21:00z"`
`DateTime midnight(DateTime myDateTime)`	Creates a DateTime object for the current midnight, relative to the specified DateTime. For example, where `MyDateTime` is `2011-05-25T17:10:00z`, the result is as follows. Example: `#{midnight(myDateTime)}` Result: `"2011-05-25T0:00:00z"`
`DateTime minusDays(DateTime myDateTime,int daysToSub)`	Creates a DateTime object that is the result of subtracting the specified number of days from the specified DateTime. Example: `#{minusDays(myDateTime,1)}` Result: `"2011-05-23T17:10:00z"`
`DateTime minusHours(DateTime myDateTime,int hoursToSub)`	Creates a DateTime object that is the result of subtracting the specified number of hours from the specified DateTime. Example: `#{minusHours(myDateTime,1)}` Result: `"2011-05-24T16:10:00z"`

Function	Description
`DateTime minusMinutes(DateTime myDateTime,int minutesToSub)`	Creates a DateTime object that is the result of subtracting the specified number of minutes from the specified DateTime. Example: `#{minusMinutes(myDateTime,1)}` Result: `"2011-05-24T17:09:00z"`
`DateTime minusMonths(DateTime myDateTime,int monthsToSub)`	Creates a DateTime object that is the result of subtracting the specified number of months from the specified DateTime. Example: `#{minusMonths(myDateTime,1)}` Result: `"2011-04-24T17:10:00z"`
`DateTime minusWeeks(DateTime myDateTime,int weeksToSub)`	Creates a DateTime object that is the result of subtracting the specified number of weeks from the specified DateTime. Example: `#{minusWeeks(myDateTime,1)}` Result: `"2011-05-17T17:10:00z"`
`DateTime minusYears(DateTime myDateTime,int yearsToSub)`	Creates a DateTime object that is the result of subtracting the specified number of years from the specified DateTime. Example: `#{minusYears(myDateTime,1)}` Result: `"2010-05-24T17:10:00z"`
`int minute(DateTime myDateTime)`	Gets the minute of the DateTime value as an integer. Example: `#{minute(myDateTime)}` Result: `10`
`int month(DateTime myDateTime)`	Gets the month of the DateTime value as an integer. Example: `#{month(myDateTime)}` Result: `5`

Function	Description
`DateTime plusDays(DateTime myDateTime,int daysToAdd)`	Creates a DateTime object that is the result of adding the specified number of days to the specified DateTime. Example: `#{plusDays(myDateTime,1)}` Result: `"2011-05-25T17:10:00z"`
`DateTime plusHours(DateTime myDateTime,int hoursToAdd)`	Creates a DateTime object that is the result of adding the specified number of hours to the specified DateTime. Example: `#{plusHours(myDateTime,1)}` Result: `"2011-05-24T18:10:00z"`
`DateTime plusMinutes(DateTime myDateTime,int minutesToAdd)`	Creates a DateTime object that is the result of adding the specified number of minutes to the specified DateTime. Example: `#{plusMinutes(myDateTime,1)}` Result: `"2011-05-24 17:11:00z"`
`DateTime plusMonths(DateTime myDateTime,int monthsToAdd)`	Creates a DateTime object that is the result of adding the specified number of months to the specified DateTime. Example: `#{plusMonths(myDateTime,1)}` Result: `"2011-06-24T17:10:00z"`
`DateTime plusWeeks(DateTime myDateTime,int weeksToAdd)`	Creates a DateTime object that is the result of adding the specified number of weeks to the specified DateTime. Example: `#{plusWeeks(myDateTime,1)}` Result: `"2011-05-31T17:10:00z"`

Function	Description
`DateTime plusYears(DateTime myDateTime,int yearsToAdd)`	Creates a DateTime object that is the result of adding the specified number of years to the specified DateTime. Example: `#{plusYears(myDateTime,1)}` Result: `"2012-05-24T17:10:00z"`
`DateTime sunday(DateTime myDateTime)`	Creates a DateTime object for the previous Sunday, relative to the specified DateTime. If the specified DateTime is a Sunday, the result is the specified DateTime. Example: `#{sunday(myDateTime)}` Result: `"2011-05-22 17:10:00 UTC"`
`int year(DateTime myDateTime)`	Gets the year of the DateTime value as an integer. Example: `#{year(myDateTime)}` Result: `2011`
`DateTime yesterday(DateTime myDateTime)`	Creates a DateTime object for the previous day, relative to the specified DateTime. The result is the same as minusDays(1). Example: `#{yesterday(myDateTime)}` Result: `"2011-05-23T17:10:00z"`

Special Characters

AWS Data Pipeline uses certain characters that have a special meaning in pipeline definitions, as shown in the following table.

Special Character	Description	Examples
@	Runtime field. This character is a field name prefix for a field that is only available when a pipeline runs.	@actualStartTime @failureReason @resourceStatus
#	Expression. Expressions are delimited by: "#{" and "}" and	#{format(myDateTime,'YYYY-MM-dd hh:mm:ss')}

Special Character	Description	Examples
	the contents of the braces are evaluated by AWS Data Pipeline. For more information, see Expressions (p. 120).	s3://mybucket/#{id}.csv
*	Encrypted field. This character is a field name prefix to indicate that AWS Data Pipeline should encrypt the contents of this field in transit between the console or CLI and the AWS Data Pipeline service.	*password

Pipeline Object Reference

You can use the following pipeline objects and components in your pipeline definition.

Contents

> **Note**
> For an example application that uses the AWS Data Pipeline Java SDK, see Data Pipeline DynamoDB Export Java Sample on GitHub.

The following is the object hierarchy for AWS Data Pipeline.

Data Nodes

The following are the AWS Data Pipeline data node objects:

Objects

DynamoDBDataNode

Defines a data node using DynamoDB, which is specified as an input to a `HiveActivity` or `EMRActivity` object.

Note
The `DynamoDBDataNode` object does not support the `Exists` precondition.

Example

The following is an example of this object type. This object references two other objects that you'd define in the same pipeline definition file. `CopyPeriod` is a `Schedule` object and `Ready` is a precondition object.

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

Syntax

Required Fields	Description	Slot Type
tableName	The Amazon DynamoDB table.	String

Object Invocation Fields	Description	Slot Type
schedule	This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), users can create a parent object that has a schedule reference. For more information about example optional schedule configurations, see http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html	Reference Object, e.g. "schedule": {"ref":"myScheduleId"}

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String

Optional Fields	Description	Slot Type
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
dataFormat	DataFormat for the data described by this data node. Currently supported for HiveActivity and HiveCopyActivity.	Reference Object, e.g. "dataFormat": {"ref":"myDynamoDBDataFormatId"}
dependsOn	Specify dependency on another runnable object	Reference Object, e.g. "dependsOn": {"ref":"myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
lateAfterTimeout	The elapsed time after pipeline start within which the object must start.	Period
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref":"myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref":"myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref":"myActionId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, e.g. "precondition": {"ref":"myPreconditionId"}
readThroughputPercent	Sets the rate of read operations to keep your DynamoDB provisioned throughput rate in the allocated range for your table. The value is a double between .1 and 1.0, inclusively.	Double
region	The code for the region where the DynamoDB table exists. For example, us-east-1. This is used by HiveActivity when it performs staging for DynamoDB tables in Hive.	Enumeration

Optional Fields	Description	Slot Type
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set then remote activities that do not report progres for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period
runsOn	The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.	Reference Object, e.g. "runsOn": {"ref":"myResourceId"}
scheduleType	Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the ActivatePipeline operation for each subsequent run. Values are: cron, ondemand, and timeseries.	Enumeration
workerGroup	The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.	String
writeThroughputPercent	Sets the rate of write operations to keep your DynamoDB provisioned throughput rate in the allocated range for your table. The value is a double between .1 and 1.0, inclusively.	Double

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of depedency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnableObjectId"}

Runtime Fields	Description	Slot Type
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String

System Fields	Description	Slot Type
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

MySqlDataNode

Defines a data node using MySQL.

Note
The `MySqlDataNode` type is deprecated. We recommend that you use SqlDataNode (p. 147) instead.

Example

The following is an example of this object type. This object references two other objects that you'd define in the same pipeline definition file. `CopyPeriod` is a `Schedule` object and `Ready` is a precondition object.

```
{
  "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" }
}
```

Syntax

Required Fields	Description	Slot Type
table	The name of the table in the MySQL database.	String

Object Invocation Fields	Description	Slot Type
schedule	This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule),	Reference Object, e.g. "schedule": {"ref":"myScheduleId"}

Object Invocation Fields	Description	Slot Type
	users can create a parent object that has a schedule reference. For more information about example optional schedule configurations, see http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html	

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
createTableSql	A SQL create table expression that will create the table	String
database	The name of the database.	Reference Object, e.g. "database": {"ref":"myDatabaseId"}
dependsOn	Specify dependency on another runnable object	Reference Object, e.g. "dependsOn": {"ref":"myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
insertQuery	A SQL statement to insert data into the table.	String
lateAfterTimeout	The elapsed time after pipeline start within which the object must start.	Period
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref":"myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref":"myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref":"myActionId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}

Optional Fields	Description	Slot Type
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, e.g. "precondition": {"ref":"myPreconditionId"}
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set then remote activities that do not report progres for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period
runsOn	The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.	Reference Object, e.g. "runsOn": {"ref":"myResourceId"}
scheduleType	Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the ActivatePipeline operation for each subsequent run. Values are: cron, ondemand, and timeseries.	Enumeration
schemaName	The name of the schema holding the table	String
selectQuery	A SQL statement to fetch data from the table.	String
workerGroup	The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.	String

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String

Runtime Fields	Description	Slot Type
@cascadeFailedOn	Description of depedency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnableObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

See Also

- S3DataNode (p. 143)

RedshiftDataNode

Defines a data node using Amazon Redshift. `RedshiftDataNode` represents the properties of the data inside a database, such as a data table, used by your pipeline.

Example

The following is an example of this object type.

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

Syntax

Required Fields	Description	Slot Type
database	The database on which the table resides	Reference Object, e.g. "database": {"ref":"myRedshiftDatabaseId"}
tableName	The name of the Amazon Redshift table. The table is created if it doesn't already exist and you've provided createTableSql.	String

Object Invocation Fields	Description	Slot Type
schedule	This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy	Reference Object, e.g. "schedule": {"ref":"myScheduleId"}

Object Invocation Fields	Description	Slot Type
	this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), users can create a parent object that has a schedule reference. For more information about example optional schedule configurations, see http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html	

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
createTableSql	A SQL expression to create the table in the database. We recommend that you specify the schema where the table should be created, for example: CREATE TABLE mySchema.myTable (bestColumn varchar(25) primary key distkey, numberOfWins integer sortKey). AWS Data Pipeline runs the script in the createTableSql field if the table, specified by tableName, does not exist in the schema, specified by the schemaName field. For example, if you specify schemaName as mySchema but do not include mySchema in the createTableSql field, the table is created in the wrong schema (by default, it would be created in PUBLIC). This occurs because AWS Data Pipeline does not parse your CREATE TABLE statements.	String
dependsOn	Specify dependency on another runnable object	Reference Object, e.g. "dependsOn": {"ref":"myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
lateAfterTimeout	The elapsed time after pipeline start within which the object must start.	Period
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure	Integer

Optional Fields	Description	Slot Type
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref":"myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref":"myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref":"myActionId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, e.g. "precondition": {"ref":"myPreconditionId"}
primaryKeys	If you do not specify primaryKeys for a destination table in RedShiftCopyActivity, you can specify a list of columns using primaryKeys which will act as a mergeKey. However, if you have an existing primaryKey defined in a Amazon Redshift table, this setting overrides the existing key.	String
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set then remote activities that do not report progres for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period
runsOn	The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.	Reference Object, e.g. "runsOn": {"ref":"myResourceId"}
scheduleType	Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the ActivatePipeline operation for each subsequent run. Values are: cron, ondemand, and timeseries.	Enumeration

Optional Fields	Description	Slot Type
schemaName	This optional field specifies the name of the schema for the Amazon Redshift table. If not specified, the schema name is PUBLIC, which is the default schema in Amazon Redshift. For more information, see the Amazon Redshift Database Developer Guide.	String
workerGroup	The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.	String

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of depedency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnableObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime

Runtime Fields	Description	Slot Type
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

S3DataNode

Defines a data node using Amazon S3. By default, the S3DataNode uses server-side encryption. If you would like to disable this, set s3EncryptionType to NONE.

> **Note**
> When you use an `S3DataNode` as input to `CopyActivity`, only the CSV and TSV data formats are supported.

Example

The following is an example of this object type. This object references another object that you'd define in the same pipeline definition file. `CopyPeriod` is a `Schedule` object.

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

Syntax

Object Invocation Fields	Description	Slot Type
schedule	This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), users can create a parent object that has a schedule reference. For more information about example optional schedule configurations, see http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html	Reference Object, e.g. "schedule": {"ref":"myScheduleId"}

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
compression	The type of compression for the data described by the S3DataNode. "none" is no compression and "gzip" is compressed with the gzip algorithm. This field is only supported for use with Amazon Redshift and when you use S3DataNode with CopyActivity.	Enumeration
dataFormat	DataFormat for the data described by this S3DataNode.	Reference Object, e.g. "dataFormat": {"ref":"myDataFormatId"}
dependsOn	Specify dependency on another runnable object	Reference Object, e.g. "dependsOn": {"ref":"myActivityId"}
directoryPath	Amazon S3 directory path as a URI: s3://my-bucket/my-key-for-directory. You must provide either a filePath or directoryPath value.	String
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
filePath	The path to the object in Amazon S3 as a URI, for example: s3://my-bucket/my-key-for-file. You must provide either a filePath or directoryPath value. Use	String

Optional Fields	Description	Slot Type
	the directoryPath value to accommodate multiple files in a directory.	
lateAfterTimeout	The elapsed time after pipeline start within which the object must start.	Period
manifestFilePath	The Amazon S3 path to a manifest file in the format supported by Amazon Redshift. AWS Data Pipeline uses the manifest file to copy the specified Amazon S3 files into the Amazon Redshift table. This field is valid only when a RedShiftCopyActivity references the S3DataNode.	String
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref":"myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref":"myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref":"myActionId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, e.g. "precondition": {"ref":"myPreconditionId"}
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set then remote activities that do not report progres for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period
runsOn	The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.	Reference Object, e.g. "runsOn": {"ref":"myResourceId"}
s3EncryptionType	Overrides the Amazon S3 encryption type. Values are SERVER_SIDE_ENCRYPTION or NONE. Server-side encryption is enabled by default.	Enumeration

Optional Fields	Description	Slot Type
scheduleType	Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the ActivatePipeline operation for each subsequent run. Values are: cron, ondemand, and timeseries.	Enumeration
workerGroup	The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.	String

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of depedency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnableObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String

Runtime Fields	Description	Slot Type
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

See Also

- MySqlDataNode (p. 135)

SqlDataNode

Defines a data node using SQL.

Example

The following is an example of this object type. This object references two other objects that you'd define in the same pipeline definition file. `CopyPeriod` is a `Schedule` object and `Ready` is a precondition object.

```
{
  "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" }
}
```

Syntax

Required Fields	Description	Slot Type
table	The name of the table in the SQL database.	String

Object Invocation Fields	Description	Slot Type
schedule	This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), users can create a parent object that has a schedule reference. For more information about example optional schedule configurations, see http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html	Reference Object, e.g. "schedule": {"ref":"myScheduleId"}

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
createTableSql	A SQL create table expression that will create the table	String

Optional Fields	Description	Slot Type
database	The name of the database.	Reference Object, e.g. "database": {"ref":"myDatabaseId"}
dependsOn	Specify dependency on another runnable object	Reference Object, e.g. "dependsOn": {"ref":"myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
insertQuery	A SQL statement to insert data into the table.	String
lateAfterTimeout	The elapsed time after pipeline start within which the object must start.	Period
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref":"myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref":"myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref":"myActionId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, e.g. "precondition": {"ref":"myPreconditionId"}
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set then remote activities that do not report progres for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period
runsOn	The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.	Reference Object, e.g. "runsOn": {"ref":"myResourceId"}

Optional Fields	Description	Slot Type
scheduleType	Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the ActivatePipeline operation for each subsequent run. Values are: cron, ondemand, and timeseries.	Enumeration
schemaName	The name of the schema holding the table	String
selectQuery	A SQL statement to fetch data from the table.	String
workerGroup	The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.	String

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of depedency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnableObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String

Runtime Fields	Description	Slot Type
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

See Also

- S3DataNode (p. 143)

Activities

The following are the AWS Data Pipeline activity objects:

Objects

- CopyActivity (p. 152)
- EmrActivity (p. 156)
- HadoopActivity (p. 162)
- HiveActivity (p. 168)
- HiveCopyActivity (p. 174)
- PigActivity (p. 179)
- RedshiftCopyActivity (p. 187)
- ShellCommandActivity (p. 194)
- SqlActivity (p. 199)

CopyActivity

Copies data from one location to another. `CopyActivity` supports S3DataNode (p. 143) and SqlDataNode (p. 147) as input and output and the copy operation is normally performed record-by-record. However, `CopyActivity` provides a high-performance Amazon S3 to Amazon S3 copy when all the following conditions are met:

- The input and output are S3DataNodes
- The `dataFormat` field is the same for input and output

If you provide compressed data files as input and do not indicate this using the `compression` field on the S3 data nodes, `CopyActivity` might fail. In this case, `CopyActivity` does not properly detect the end of record character and the operation fails. Further, `CopyActivity` supports copying from a directory to another directory and copying a file to a directory, but record-by-record copy occurs when copying a directory to a file. Finally, `CopyActivity` does not support copying multipart Amazon S3 files.

`CopyActivity` has specific limitations to its CSV support. When you use an S3DataNode as input for `CopyActivity`, you can only use a Unix/Linux variant of the CSV data file format for the Amazon S3 input and output fields. The Unix/Linux variant requires the following:

- The separator must be the "," (comma) character.
- The records are not quoted.
- The default escape character is ASCII value 92 (backslash).
- The end of record identifier is ASCII value 10 (or "\n").

Windows-based systems typically use a different end-of-record character sequence: a carriage return and line feed together (ASCII value 13 and ASCII value 10). You must accommodate this difference using an additional mechanism, such as a pre-copy script to modify the input data, to ensure that `CopyActivity` can properly detect the end of a record; otherwise, the `CopyActivity` fails repeatedly.

When using `CopyActivity` to export from a PostgreSQL RDS object to a TSV data format, the default NULL character is \n.

Example

The following is an example of this object type. This object references three other objects that you would define in the same pipeline definition file. `CopyPeriod` is a `Schedule` object and `InputData` and `OutputData` are data node objects.

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

Syntax

Object Invocation Fields	Description	Slot Type
schedule	This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), users can create a parent object that has a schedule reference. For more information about example optional schedule configurations, see http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html	Reference Object, e.g. "schedule": {"ref":"myScheduleId"}

Required Group (One of the following is required)	Description	Slot Type
runsOn	The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.	Reference Object, e.g. "runsOn": {"ref":"myResourceId"}
workerGroup	The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.	String

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String

Optional Fields	Description	Slot Type
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
dependsOn	Specify dependency on another runnable object.	Reference Object, e.g. "dependsOn": {"ref":"myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
input	The input data source.	Reference Object, e.g. "input": {"ref":"myDataNodeId"}
lateAfterTimeout	The elapsed time after pipeline start within which the object must start.	Period
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref":"myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref":"myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref":"myActionId"}
output	The output data source.	Reference Object, e.g. "output": {"ref":"myDataNodeId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, e.g. "precondition": {"ref":"myPreconditionId"}
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set then remote activities that do not report progres for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period

Optional Fields	Description	Slot Type
scheduleType	Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the ActivatePipeline operation for each subsequent run. Values are: cron, ondemand, and timeseries.	Enumeration

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of depedency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnableObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	The id of the last instance object that reached a terminated state.	String

Runtime Fields	Description	Slot Type
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

See Also

- ShellCommandActivity (p. 194)
- EmrActivity (p. 156)
- Export MySQL Data to Amazon S3 Using AWS Data Pipeline (p. 96)

EmrActivity

Runs an EMR cluster.

AWS Data Pipeline uses a different format for steps than Amazon EMR; for example, AWS Data Pipeline uses comma-separated arguments after the JAR name in the `EmrActivity` step field. The following example shows a step formatted for Amazon EMR, followed by its AWS Data Pipeline equivalent:

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

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

Examples

The following is an example of this object type. This object references three other objects that you would define in the same pipeline definition file. `MyEmrCluster` is an `EmrCluster` object and `MyS3Input` and `MyS3Output` are `S3DataNode` objects.

Note
In this example, you can replace the `step` field with your desired cluster string, which could be a Pig script, Hadoop streaming cluster, your own custom JAR including its parameters, or so on.

Hadoop 1.x (AMI 2.x)

```
{
  "id" : "MyEmrActivity",
  "type" : "EmrActivity",
  "runsOn" : { "ref" : "MyEmrCluster" },
  "preStepCommand" : "scp remoteFiles localFiles",
  "step" : ["s3://mybucket/myPath/myStep.jar,firstArg,secondArg","s3://mybucket/myPath/
myOtherStep.jar,anotherArg"],
  "postStepCommand" : "scp localFiles remoteFiles",
  "input" : { "ref" : "MyS3Input" },
  "output" : { "ref" : "MyS3Output" }
}
```

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
To pass arguments to an application in a step, you may need to escape the arguments that you pass. For example, if you use `script-runner.jar` to run a shell script and want to pass arguments to the script, you must escape the commas that separate them. The following step slot illustrates how to do this:

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

This step uses `script-runner.jar` to run the `echo.sh` shell script and passes `a`, `b`, and `c` as a single argument to the script. The first escape character is removed from the resultant argument

so you may need to escape again. For example, if you had `File\.gz` as an argument in JSON, you could escape it using `File\\\\.gz`. However, because the first escape is discarded, you must use `File\\\\\\\\.gz` .

Syntax

Object Invocation Fields	Description	Slot Type
schedule	This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), users can create a parent object that has a schedule reference. For more information about example optional schedule configurations, see http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html	Reference Object, e.g. "schedule": {"ref":"myScheduleId"}

Required Group (One of the following is required)	Description	Slot Type
runsOn	EMR Cluster on which this job will run.	Reference Object, e.g. "runsOn": {"ref":"myEmrClusterId"}
workerGroup	The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.	String

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
dependsOn	Specify dependency on another runnable object.	Reference Object, e.g. "dependsOn": {"ref":"myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration

Optional Fields	Description	Slot Type
input	Location of the input data.	Reference Object, e.g. "input": {"ref":"myDataNodeId"}
lateAfterTimeout	The elapsed time after pipeline start within which the object must start.	Period
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref":"myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref":"myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref":"myActionId"}
output	Location of the output data.	Reference Object, e.g. "output": {"ref":"myDataNodeId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
postStepCommand	Shell scripts to be run after all steps are finished. To specify multiple scripts, up to 255, add multiple postStepCommand fields.	String
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, e.g. "precondition": {"ref":"myPreconditionId"}
preStepCommand	Shell scripts to be run before any steps are run. To specify multiple scripts, up to 255, add multiple preStepCommand fields.	String
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set then remote activities that do not report progres for the specified period may be considered stalled and so retried.	Period
resizeClusterBeforeRunning	Resize the cluster before performing this activity to accomodate DynamoDB tables specified as inputs or outputs	Boolean

Optional Fields	Description	Slot Type
resizeClusterMaxInstances	A limit on the maximum number of instance that can be requested by the resize algorithm	Integer
retryDelay	The timeout duration between two retry attempts.	Period
scheduleType	Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the ActivatePipeline operation for each subsequent run. Values are: cron, ondemand, and timeseries.	Enumeration
step	One or more steps for the cluster to run. To specify multiple steps, up to 255, add multiple step fields. Use comma-separated arguments after the JAR name; for example, "s3://example-bucket/MyWork.jar,arg1,arg2,arg3".	String

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of depedency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnableObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime

Runtime Fields	Description	Slot Type
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

See Also

- ShellCommandActivity (p. 194)
- CopyActivity (p. 152)

- EmrCluster (p. 210)

HadoopActivity

Runs a MapReduce job on a cluster. The cluster can be an EMR cluster managed by AWS Data Pipeline or another resource if you use TaskRunner. Use HadoopActivity when you want to run work in parallel. This allows you to use the scheduling resources of the YARN framework or the MapReduce resource negotiator in Hadoop 1. If you would like to run work sequentially using the Amazon EMR Step action, you can still use EmrActivity (p. 156).

Examples

HadoopActivity using an EMR cluster managed by AWS Data Pipeline

The following HadoopActivity object uses an EmrCluster resource to run a program:

```
 {
   "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”
 }
```

Here is the corresponding *MyEmrCluster*, which configures the FairScheduler and queues in YARN for Hadoop 2-based AMIs:

```
{
  "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”]
}
```

This is the EmrCluster you use to configure FairScheduler in Hadoop 1:

```
{
      "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"
          }
```

The following EmrCluster configures CapacityScheduler for Hadoop 2-based AMIs:

```
{
      "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"
    }
```

HadoopActivity using an existing EMR cluster

In this example, you use workergroups and a TaskRunner to run a program on an existing EMR cluster. The following pipeline definition uses HadoopActivity to:

- Run a MapReduce program only on *myWorkerGroup* resources. For more information about worker groups, see Executing Work on Existing Resources Using Task Runner (p. 264).
- Run a preActivityTaskConfig and 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"
      }
    }
  ]
}
```

Syntax

Required Fields	Description	Slot Type
jarUri	Location of a JAR in Amazon S3 or the local file system of the cluster to run with HadoopActivity.	String

Object Invocation Fields	Description	Slot Type
schedule	This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), users can create a parent object that has a	Reference Object, e.g. "schedule": {"ref":"myScheduleId"}

Object Invocation Fields	Description	Slot Type
	schedule reference. For more information about example optional schedule configurations, see http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html	

Required Group (One of the following is required)	Description	Slot Type
runsOn	EMR Cluster on which this job will run.	Reference Object, e.g. "runsOn": {"ref":"myEmrClusterId"}
workerGroup	The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.	String

Optional Fields	Description	Slot Type
argument	Arguments to pass to the JAR.	String
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
dependsOn	Specify dependency on another runnable object.	Reference Object, e.g. "dependsOn": {"ref":"myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
hadoopQueue	The Hadoop scheduler queue name on which the activity will be submitted.	String
input	Location of the input data.	Reference Object, e.g. "input": {"ref":"myDataNodeId"}
lateAfterTimeout	The elapsed time after pipeline start within which the object must start.	Period
mainClass	The main class of the JAR you are executing with HadoopActivity.	String
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure	Integer

Optional Fields	Description	Slot Type
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref":"myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref":"myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref":"myActionId"}
output	Location of the output data.	Reference Object, e.g. "output": {"ref":"myDataNodeId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
postActivityTaskConfig	Post-activity configuration script to be run. This consists of a URI of the shell script in Amazon S3 and a list of arguments.	Reference Object, e.g. "postActivityTaskConfig": {"ref":"myShellScriptConfigId"}
preActivityTaskConfig	Pre-activity configuration script to be run. This consists of a URI of the shell script in Amazon S3 and a list of arguments.	Reference Object, e.g. "preActivityTaskConfig": {"ref":"myShellScriptConfigId"}
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, e.g. "precondition": {"ref":"myPreconditionId"}
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set then remote activities that do not report progres for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period

Optional Fields	Description	Slot Type
scheduleType	Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the ActivatePipeline operation for each subsequent run. Values are: cron, ondemand, and timeseries.	Enumeration

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of depedency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnableObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	The id of the last instance object that reached a terminated state.	String

Runtime Fields	Description	Slot Type
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

See Also

- ShellCommandActivity (p. 194)
- CopyActivity (p. 152)
- EmrCluster (p. 210)

HiveActivity

Runs a Hive query on an EMR cluster. `HiveActivity` makes it easier to set up an Amazon EMR activity and automatically creates Hive tables based on input data coming in from either Amazon S3 or Amazon RDS. All you need to specify is the HiveQL to run on the source data. AWS Data Pipeline automatically creates Hive tables with `${input1}`, `${input2}`, and so on, based on the input fields in the HiveActivity

object. For Amazon S3 inputs, the `dataFormat` field is used to create the Hive column names. For MySQL (Amazon RDS) inputs, the column names for the SQL query are used to create the Hive column names.

Note
This activity uses the Hive CSV Serde.

Example

The following is an example of this object type. This object references three other objects that you would define in the same pipeline definition file. `MySchedule` is a `Schedule` object and `MyS3Input` and `MyS3Output` are data node objects.

```
{
  "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" }
}
```

Syntax

Object Invocation Fields	Description	Slot Type
schedule	This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), users can create a parent object that has a schedule reference. For more information about example optional schedule configurations, see http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html	Reference Object, e.g. "schedule": {"ref":"myScheduleId"}

Required Group (One of the following is required)	Description	Slot Type
hiveScript	The Hive script to run.	String
scriptUri	The location of the Hive script to run (for example, s3://scriptLocation).	String

Required Group (One of the following is required)	Description	Slot Type
runsOn	EMR Cluster on which this HiveActivity runs.	Reference Object, e.g. "runsOn": {"ref":"myEmrClusterId"}
workerGroup	The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.	String

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
dependsOn	Specify dependency on another runnable object.	Reference Object, e.g. "dependsOn": {"ref":"myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
hadoopQueue	The Hadoop scheduler queue name on which the job will be submitted.	String
input	The input data source.	Reference Object, e.g. "input": {"ref":"myDataNodeId"}
lateAfterTimeout	The elapsed time after pipeline start within which the object must start.	Period
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref":"myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref":"myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref":"myActionId"}

Optional Fields	Description	Slot Type
output	The output data source.	Reference Object, e.g. "output": {"ref":"myDataNodeId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
postActivityTaskConfig	Post-activity configuration script to be run. This consists of a URI of the shell script in Amazon S3 and a list of arguments.	Reference Object, e.g. "postActivityTaskConfig": {"ref":"myShellScriptConfigId"}
preActivityTaskConfig	Pre-activity configuration script to be run. This consists of a URI of the shell script in Amazon S3 and a list of arguments.	Reference Object, e.g. "preActivityTaskConfig": {"ref":"myShellScriptConfigId"}
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, e.g. "precondition": {"ref":"myPreconditionId"}
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set then remote activities that do not report progres for the specified period may be considered stalled and so retried.	Period
resizeClusterBeforeRunning	Resize the cluster before performing this activity to accomodate DynamoDB tables specified as inputs or outputs	Boolean
resizeClusterMaxInstances	A limit on the maximum number of instance that can be requested by the resize algorithm	Integer
retryDelay	The timeout duration between two retry attempts.	Period
scheduleType	Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the ActivatePipeline operation for each subsequent run. Values are: cron, ondemand, and timeseries.	Enumeration

Optional Fields	Description	Slot Type
scriptVariable	Specifies script variables for Amazon EMR to pass to Hive while running a script. For example, the following example script variables would pass a SAMPLE and FILTER_DATE variable to Hive: SAMPLE=s3://elasticmapreduce/samples/hive-ads and FILTER_DATE=#{format(@scheduledStartTime,'YYYY-MM-dd')}% This field accepts multiple values and works with both script and scriptUri fields. In addition, scriptVariable functions regardless of whether stage is set to true or false. This field is especially useful to send dynamic values to Hive using AWS Data Pipeline expressions and functions.	String
stage	Determines whether staging is enabled before or after running the script. Not permitted with Hive 11, so use an Amazon EMR AMI version 3.2.0 or greater.	Boolean

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of depedency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnableObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String

Runtime Fields	Description	Slot Type
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

See Also

- ShellCommandActivity (p. 194)
- EmrActivity (p. 156)

HiveCopyActivity

Runs a Hive query on an EMR cluster. `HiveCopyActivity` makes it easier to copy data between DynamoDB tables. `HiveCopyActivity` accepts a HiveQL statement to filter input data from DynamoDB at the column and row level.

Example

The following example shows how to use `HiveCopyActivity` and `DynamoDBExportDataFormat` to copy data from one `DynamoDBDataNode` to another, while filtering data, based on a time stamp.

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

Syntax

Object Invocation Fields	Description	Slot Type
schedule	This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), users can create a parent object that has a schedule reference. For more information about example optional schedule configurations, see http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html	Reference Object, e.g. "schedule": {"ref":"myScheduleId"}

Required Group (One of the following is required)	Description	Slot Type
runsOn	Specify cluster to run on.	Reference Object, e.g. "runsOn": {"ref":"myResourceId"}
workerGroup	The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.	String

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
dependsOn	Specify dependency on another runnable object.	Reference Object, e.g. "dependsOn": {"ref":"myActivityId"}

Optional Fields	Description	Slot Type
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
filterSql	A Hive SQL statement fragment that filters a subset of DynamoDB or Amazon S3 data to copy. The filter should only contain predicates and not begin with a WHERE clause, because AWS Data Pipeline adds it automatically.	String
input	The input data source. This must be a S3DataNode or DynamoDBDataNode. If you use DynamoDBNode, specify a DynamoDBExportDataFormat.	Reference Object, e.g. "input": {"ref":"myDataNodeId"}
lateAfterTimeout	The elapsed time after pipeline start within which the object must start.	Period
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref":"myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref":"myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref":"myActionId"}
output	The output data source. If input is S3DataNode, this must be DynamoDBDataNode. Otherwise, this can be S3DataNode or DynamoDBDataNode. If you use DynamoDBNode, specify a DynamoDBExportDataFormat.	Reference Object, e.g. "output": {"ref":"myDataNodeId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
postActivityTaskConfig	Post-activity configuration script to be run. This consists of a URI of the shell script in Amazon S3 and a list of arguments.	Reference Object, e.g. "postActivityTaskConfig": {"ref":"myShellScriptConfigId"}
preActivityTaskConfig	Pre-activity configuration script to be run. This consists of a URI of the shell script in Amazon S3 and a list of arguments.	Reference Object, e.g. "preActivityTaskConfig": {"ref":"myShellScriptConfigId"}

Optional Fields	Description	Slot Type
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, e.g. "precondition": {"ref":"myPreconditionId"}
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set then remote activities that do not report progres for the specified period may be considered stalled and so retried.	Period
resizeClusterBeforeRunning	Resize the cluster before performing this activity to accomodate DynamoDB tables specified as inputs or outputs	Boolean
resizeClusterMaxInstances	A limit on the maximum number of instance that can be requested by the resize algorithm	Integer
retryDelay	The timeout duration between two retry attempts.	Period
scheduleType	Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the ActivatePipeline operation for each subsequent run. Values are: cron, ondemand, and timeseries.	Enumeration

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of depedency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnableObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String

Runtime Fields	Description	Slot Type
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String

System Fields	Description	Slot Type
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

See Also

- ShellCommandActivity (p. 194)
- EmrActivity (p. 156)

PigActivity

PigActivity provides native support for Pig scripts in AWS Data Pipeline without the requirement to use `ShellCommandActivity` or `EmrActivity`. In addition, PigActivity supports data staging. When the stage field is set to true, AWS Data Pipeline stages the input data as a schema in Pig without additional code from the user.

Example

The following example pipeline shows how to use `PigActivity`. The example pipeline performs the following steps:

- MyPigActivity1 loads data from Amazon S3 and runs a Pig script that selects a few columns of data and uploads it to Amazon S3.
- MyPigActivity2 loads the first output, selects a few columns and three rows of data, and uploads it to Amazon S3 as a second output.
- MyPigActivity3 loads the second output data, inserts two rows of data and only the column named "fifth" to Amazon RDS.
- MyPigActivity4 loads Amazon RDS data, selects the first row of data, and uploads it to 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"
    }
  ]
}
```

The content of `pigTestScript.q` is as follows.

```
B = LIMIT ${input1} $three; ${output1} = FOREACH B GENERATE $column1, $column2, Third,
 Fourth, Fifth, Sixth, Seventh, Eighth;
```

Syntax

Object Invocation Fields	Description	Slot Type
schedule	This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), users can create a parent object that has a schedule reference. For more information about example optional schedule configurations, see http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html	Reference Object, e.g. "schedule": {"ref":"myScheduleId"}

Required Group (One of the following is required)	Description	Slot Type
script	The Pig script to run.	String

Required Group (One of the following is required)	Description	Slot Type
scriptUri	The location of the Pig script to run (for example, s3://scriptLocation).	String

Required Group (One of the following is required)	Description	Slot Type
runsOn	EMR Cluster on which this PigActivity runs.	Reference Object, e.g. "runsOn": {"ref":"myEmrClusterId"}
workerGroup	The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.	String

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
dependsOn	Specify dependency on another runnable object.	Reference Object, e.g. "dependsOn": {"ref":"myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
input	The input data source.	Reference Object, e.g. "input": {"ref":"myDataNodeId"}
lateAfterTimeout	The elapsed time after pipeline start within which the object must start.	Period
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref":"myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref":"myActionId"}

Optional Fields	Description	Slot Type
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref":"myActionId"}
output	The output data source.	Reference Object, e.g. "output": {"ref":"myDataNodeId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
postActivityTaskConfig	Post-activity configuration script to be run. This consists of a URI of the shell script in Amazon S3 and a list of arguments.	Reference Object, e.g. "postActivityTaskConfig": {"ref":"myShellScriptConfigId"}
preActivityTaskConfig	Pre-activity configuration script to be run. This consists of a URI of the shell script in Amazon S3 and a list of arguments.	Reference Object, e.g. "preActivityTaskConfig": {"ref":"myShellScriptConfigId"}
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, e.g. "precondition": {"ref":"myPreconditionId"}
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set then remote activities that do not report progres for the specified period may be considered stalled and so retried.	Period
resizeClusterBeforeRunning	Resize the cluster before performing this activity to accomodate DynamoDB tables specified as inputs or outputs	Boolean
resizeClusterMaxInstances	A limit on the maximum number of instance that can be requested by the resize algorithm	Integer
retryDelay	The timeout duration between two retry attempts.	Period
scheduleType	Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the ActivatePipeline operation for each subsequent run. Values are: cron, ondemand, and timeseries.	Enumeration

Optional Fields	Description	Slot Type
scriptVariable	The arguments to pass to the Pig script. You can use scriptVariable with script or scriptUri.	String
stage	Determines whether staging is enabled and allows your Pig script to have access to the staged-data tables, such as ${INPUT1} and ${OUTPUT1}.	Boolean

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of depedency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnableObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime

Runtime Fields	Description	Slot Type
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

See Also

- ShellCommandActivity (p. 194)
- EmrActivity (p. 156)

RedshiftCopyActivity

Copies data from DynamoDB or Amazon S3 to Amazon Redshift. You can load data into a new table, or easily merge data into an existing table.

You can also move data from Amazon RDS and Amazon EMR to Amazon Redshift by using AWS Data Pipeline to stage your data in Amazon S3 before loading it into Amazon Redshift to analyze it. In addition, `RedshiftCopyActivity` supports a manifest file when working with an `S3DataNode`. You can also copy from Amazon Redshift to Amazon S3 using `RedshiftCopyActivity`. For more information, see S3DataNode (p. 143).

You can use SqlActivity (p. 199) to perform SQL queries on the data that you've loaded into Amazon Redshift.

Example

The following is an example of this object type.

```
{
  "id" : "S3ToRedshiftCopyActivity",
  "type" : "RedshiftCopyActivity",
  "input" : { "ref": "MyS3DataNode" },
  "output" : { "ref": "MyRedshiftDataNode" },
  "insertMode" : "KEEP_EXISTING",
  "schedule" : { "ref": "Hour" },
  "runsOn" : { "ref": "MyEc2Resource" },
  "commandOptions": ["EMPTYASNULL", "IGNOREBLANKLINES"]
}
```

The following example pipeline definition shows an activity that uses the APPEND insert mode:

```
{
  "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 operation adds items to a table regardless of the primary or sort keys. For example, if you have the following table, you can append a record with the same ID and user value.

```
ID(PK)     USER
1          aaa
2          bbb
```

You can append a record with the same ID and user value:

```
ID(PK)     USER
1          aaa
2          bbb
1          aaa
```

Note

With an APPEND operation that is interrupted and retried, the resulting rerun pipeline potentially appends from the beginning. This may cause further duplication, so you should be aware of this behavior, especially if you have any logic that counts the number of rows.

For a tutorial, see Copy Data to Amazon Redshift Using AWS Data Pipeline (p. 107).

Syntax

Required Fields	Description	Slot Type
insertMode	Determines what AWS Data Pipeline does with pre-existing data in the target table that overlaps with rows in the data to be loaded. Valid values are KEEP_EXISTING, OVERWRITE_EXISTING, TRUNCATE and APPEND. KEEP_EXISTING adds new rows to the table, while leaving any existing rows unmodified. KEEP_EXISTING and OVERWRITE_EXISTING use the primary key, sort, and distribution keys to identify which incoming rows to match with existing rows, according to the information provided in Updating and inserting new data in the Amazon Redshift Database Developer Guide. TRUNCATE deletes all the data in the destination table before writing the new data. APPEND will add all records to the end of the Redshift table. APPEND does not require a primary, distribution key, or sort key so items that may be potential duplicates may be appended.	Enumeration

Object Invocation Fields	Description	Slot Type
schedule	This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), users can create a parent object that has a schedule reference. For more information about example optional schedule configurations, see http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html	Reference Object, e.g. "schedule": {"ref":"myScheduleId"}

Required Group (One of the following is required)	Description	Slot Type
runsOn	The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.	Reference Object, e.g. "runsOn": {"ref":"myResourceId"}

Required Group (One of the following is required)	Description	Slot Type
workerGroup	The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.	String

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
commandOptions	Takes parameters to pass to the Amazon Redshift data node during the COPY operation. For information about Amazon Redshift COPY parameters, see the Parameters section in the COPY topic in the Amazon Redshift Database Developer Guide. If a data format is associated with the input or output data node, then the provided parameters are ignored. Because the copy operation first uses COPY to insert data into a staging table, and then uses an INSERT command to copy the data from the staging table into the destination table, some COPY parameters do not apply, such as the COPY command's ability to enable automatic compression of the table. If compression is required, add column encoding details to the CREATE TABLE statement.	String
dependsOn	Specify dependency on another runnable object.	Reference Object, e.g. "dependsOn": {"ref":"myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
input	The input data node. The data source can be Amazon S3, DynamoDB, or Amazon Redshift.	Reference Object, e.g. "input": {"ref":"myDataNodeId"}
lateAfterTimeout	The elapsed time after pipeline start within which the object must start.	Period
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref":"myActionId"}

Optional Fields	Description	Slot Type
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref":"myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref":"myActionId"}
output	The output data node. The output location can be Amazon S3 or Amazon Redshift.	Reference Object, e.g. "output": {"ref":"myDataNodeId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, e.g. "precondition": {"ref":"myPreconditionId"}
queue	Corresponds to the query_group setting in Amazon Redshift, which allows you to assign and prioritize concurrent activities based on their placement in queues. Amazon Redshift limits the number of simultaneous connections to 15. For more information, see Assigning Queries to Queues in the Amazon Redshift Database Developer Guide.	String
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set then remote activities that do not report progres for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period
scheduleType	Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the ActivatePipeline operation for each subsequent run. Values are: cron, ondemand, and timeseries.	Enumeration

Optional Fields	Description	Slot Type
transformSql	The SQL SELECT expression used to transform the input data. When you copy data from DynamoDB or Amazon S3, AWS Data Pipeline creates a table called staging and initially loads it in there. Data from this table is used to update the target table. If the transformSql option is specified, a second staging table is created from the specified SQL statement. The data from this second staging table is then updated in the final target table. transformSql must be run on the table named staging and the output schema of transformSql must match the final target table's schema.	String

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of depedency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnableObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	The id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String

Runtime Fields	Description	Slot Type
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

ShellCommandActivity

Runs a command or script. You can use `ShellCommandActivity` to run time-series or cron-like scheduled tasks.

When the `stage` field is set to true and used with an `S3DataNode`, `ShellCommandActivity` supports the concept of staging data, which means that you can move data from Amazon S3 to a stage location, such as Amazon EC2 or your local environment, perform work on the data using scripts and the `ShellCommandActivity`, and move it back to Amazon S3. In this case, when your shell command is connected to an input S3DataNode, your shell scripts operate directly on the data using `${INPUT1_STAGING_DIR}`, `${INPUT2_STAGING_DIR}`, etc. referring to the `ShellCommandActivity` input fields. Similarly, output from the shell-command can be staged in an output directory to be automatically pushed to Amazon S3, referred to by `${OUTPUT1_STAGING_DIR}`, `${OUTPUT2_STAGING_DIR}`, and so on. These expressions can pass as command-line arguments to the shell-command for you to use in data transformation logic.

`ShellCommandActivity` returns Linux-style error codes and strings. If a `ShellCommandActivity` results in error, the `error` returned will be a non-zero value.

Example

The following is an example of this object type.

```
{
  "id" : "CreateDirectory",
  "type" : "ShellCommandActivity",
  "command" : "mkdir new-directory"
}
```

Syntax

Object Invocation Fields	Description	Slot Type
schedule	This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), users can create a parent object that has a schedule reference. For more information about example optional schedule configurations, see http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html	Reference Object, e.g. "schedule": {"ref":"myScheduleId"}

Required Group (One of the following is required)	Description	Slot Type
command	The command to run. Use $ to reference positional parameters and scriptArgument to specify the parameters for the command. This value and any associated parameters must function in the environment from which you are running the Task Runner.	String
scriptUri	An Amazon S3 URI path for a file to download and run as a shell command. Only one scriptUri or command field should be present. scriptUri cannot use parameters, use command instead.	String

Required Group (One of the following is required)	Description	Slot Type
runsOn	The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.	Reference Object, e.g. "runsOn": {"ref":"myResourceId"}
workerGroup	The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.	String

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
dependsOn	Specify dependency on another runnable object.	Reference Object, e.g. "dependsOn": {"ref":"myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
input	Location of the input data.	Reference Object, e.g. "input": {"ref":"myDataNodeId"}
lateAfterTimeout	The elapsed time after pipeline start within which the object must start.	Period
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref":"myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref":"myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref":"myActionId"}
output	Location of the output data.	Reference Object, e.g. "output": {"ref":"myDataNodeId"}

Optional Fields	Description	Slot Type
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, e.g. "precondition": {"ref":"myPreconditionId"}
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set then remote activities that do not report progres for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period
scheduleType	Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the ActivatePipeline operation for each subsequent run. Values are: cron, ondemand, and timeseries.	Enumeration
scriptArgument	A JSON-formatted array of strings to pass to the command specified by command. For example, if command is echo $1 $2, you can specify scriptArgument as "param1", "param2". If you had multiple arguments and parameters, you can pass scriptArgument as follows: "scriptArgument":"arg1","scriptArgument":"param1","scriptArgument":"arg2","scriptArgument": scriptArgument can only be used with command, and using with scriptUri causes an error.	String
stage	Determines whether staging is enabled and allows your shell commands to have access to the staged-data variables, such as ${INPUT1_STAGING_DIR} and ${OUTPUT1_STAGING_DIR}.	Boolean
stderr	The path that receives redirected system error messages from the command. If you use the runsOn field, this must be an Amazon S3 path because of the transitory nature of the resource running your activity. However if you specify the workerGroup field, a local file path is permitted.	String

Optional Fields	Description	Slot Type
stdout	The Amazon S3 path that receives redirected output from the command. If you use the runsOn field, this must be an Amazon S3 path because of the transitory nature of the resource running your activity. However if you specify the workerGroup field, a local file path is permitted.	String

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of depedency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnableObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime

Runtime Fields	Description	Slot Type
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

See Also

- CopyActivity (p. 152)
- EmrActivity (p. 156)

SqlActivity

Runs an SQL query (script) on a database.

Example

The following is an example of this object type.

```
{
  "id" : "MySqlActivity",
  "type" : "SqlActivity",
  "database" : { "ref": "MyDatabaseID" },
  "script" : "SQLQuery" | "scriptUri" : s3://scriptBucket/query.sql,
  "schedule" : { "ref": "MyScheduleID" },
}
```

Syntax

Required Fields	Description	Slot Type
database	The database on which to execute the supplied SQL script	Reference Object, e.g. "database": {"ref":"myDatabaseId"}

Object Invocation Fields	Description	Slot Type
schedule	This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), users can create a parent object that has a schedule reference. For more information about example optional schedule configurations, see http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html	Reference Object, e.g. "schedule": {"ref":"myScheduleId"}

Required Group (One of the following is required)	Description	Slot Type
script	The SQL script to run. You must specify script or scriptUri. When the script is stored in Amazon S3, then script is not evaluated as an expression. Specifying multiple values for scriptArgument is helpful when the script is stored in Amazon S3.	String
scriptUri	A URI specifying the location of a SQL script to execute in this activity	String

Required Group (One of the following is required)	Description	Slot Type
runsOn	The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.	Reference Object, e.g. "runsOn": {"ref":"myResourceId"}

Required Group (One of the following is required)	Description	Slot Type
workerGroup	The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.	String

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
dependsOn	Specify dependency on another runnable object.	Reference Object, e.g. "dependsOn": {"ref":"myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
input	Location of the input data.	Reference Object, e.g. "input": {"ref":"myDataNodeId"}
lateAfterTimeout	The time period since the scheduled start of the pipeline within which the object run must start.	Period
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref":"myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed in the time period since the scheduled start of the pipeline as specified by 'lateAfterTimeout'.	Reference Object, e.g. "onLateAction": {"ref":"myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref":"myActionId"}
output	Location of the output data. This is only useful for referencing from within a script (for example `#{output.tablename}`) and for creating the output table by setting 'createTableSql' in the output data node. The output of the SQL query is not written to the output data node.	Reference Object, e.g. "output": {"ref":"myDataNodeId"}

Optional Fields	Description	Slot Type
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, e.g. "precondition": {"ref":"myPreconditionId"}
queue	[Amazon Redshift only] Corresponds to the query_group setting in Amazon Redshift, which allows you to assign and prioritize concurrent activities based on their placement in queues. Amazon Redshift limits the number of simultaneous connections to 15. For more information, see Assigning Queries to Queues in the Amazon Redshift Database Developer Guide.	String
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set then remote activities that do not report progres for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period
scheduleType	Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the ActivatePipeline operation for each subsequent run. Values are: cron, ondemand, and timeseries.	Enumeration
scriptArgument	A list of variables for the script. You can alternatively put expressions directly into the script field. Multiple values for scriptArgument are helpful when the script is stored in Amazon S3. Example: #{format(@scheduledStartTime, "YY-MM-DD HH:MM:SS"}\n#{format(plusPeriod(@scheduledStartTime, "1 day"), "YY-MM-DD HH:MM:SS"}	String

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of depedency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnableObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime

Runtime Fields	Description	Slot Type
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

Resources

The following are the AWS Data Pipeline resource objects:

Objects

- Ec2Resource (p. 204)
- EmrCluster (p. 210)
- HttpProxy (p. 221)

Ec2Resource

An EC2 instance that performs the work defined by a pipeline activity.

Examples

EC2-Classic

The following example object launches an EC2 instance into EC2-Classic or a default VPC, with some optional fields set.

```
{
  "id" : "MyEC2Resource",
  "type" : "Ec2Resource",
  "actionOnTaskFailure" : "terminate",
  "actionOnResourceFailure" : "retryAll",
  "maximumRetries" : "1",
  "instanceType" : "m1.medium",
  "securityGroups" : [
    "test-group",
    "default"
  ],
```

```
  "keyPair" : "my-key-pair"
}
```

EC2-VPC

The following example object launches an EC2 instance into a nondefault VPC, with some optional fields set.

```
{
  "id" : "MyEC2Resource",
  "type" : "Ec2Resource",
  "actionOnTaskFailure" : "terminate",
  "actionOnResourceFailure" : "retryAll",
  "maximumRetries" : "1",
  "instanceType" : "m1.medium",
  "securityGroupIds" : [
    "sg-12345678",
    "sg-12345678"
  ],
  "subnetId": "subnet-12345678",
  "associatePublicIpAddress": "true",
  "keyPair" : "my-key-pair"
}
```

Syntax

Required Fields	Description	Slot Type
resourceRole	The IAM role that controls the resources that the EC2 instance can access.	String
role	The IAM role that AWS Data Pipeline uses to create the EC2 instance.	String

Object Invocation Fields	Description	Slot Type
schedule	This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), users can create a parent object that has a schedule reference. For more information about example optional schedule configurations, see http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html	Reference Object, e.g. "schedule": {"ref":"myScheduleId"}

Optional Fields	Description	Slot Type
actionOnResourceFailure	The action taken after a resource failure for this resource. Valid values are "retryall" and "retrynone".	String
actionOnTaskFailure	The action taken after task failure for this resource. Valid values are "continue" or "terminate".	String
associatePublicIpAddress	Indicates whether to assign a public IP address to the instance. If the instance is in EC2-Classic or a default VPC, the default value is true. Otherwise, the default value is false.	Boolean
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
availabilityZone	The Availability Zone in which to launch the EC2 instance.	String
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
httpProxy	The proxy host that clients use to connect to AWS services.	Reference Object, e.g. "httpProxy": {"ref":"myHttpProxyId"}
imageId	The ID of the AMI to use for the instance. By default, AWS Data Pipeline uses the PV AMI virtualization type. The specific AMI IDs used are based on region as follows: • us-east-1: ami-05355a6c • us-west-1: ami-3ffed17a • us-west-2: ami-0358ce33 • eu-west-1: ami-c7c0d6b3 • ap-southeast-1: ami-fade91a8 • ap-southeast-2: ami-d16bfbeb • ap-northeast-1: ami-39b23d38 • sa-east-1: ami-5253894f • eu-cental-1: ami-b43503a9 If the `instanceType` specified does not support PV AMIs (see Amazon Linux AMI Instance Type Matrix, specify the ID of an HVM AMI or an error occurs. For more information about AMI types, see Linux AMI Virtualization Types and Finding a Linux AMI in the *Amazon EC2 User Guide for Linux Instances*.	String
initTimeout	The amount of time to wait for the resource to start.	Period

Optional Fields	Description	Slot Type
instanceCount	Deprecated	Integer
instanceType	The type of EC2 instance to start.	String
keyPair	The name of the key pair. If you launch an EC2 instance without specifying a key pair, you can't log on to it.	String
lateAfterTimeout	The elapsed time after pipeline start within which the object must start.	Period
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure	Integer
minInstanceCount	Deprecated	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref":"myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref":"myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref":"myActionId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
region	The code for the region that the EC2 instance should run in. By default, the instance runs in the same region as the pipeline. You can run the instance in the same region as a dependent data set.	Enumeration
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set then remote activities that do not report progres for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period
runAsUser	The user to run TaskRunner	String
runsOn	This Field is not allowed on this object.	Reference Object, e.g. "runsOn": {"ref":"myResourceId"}

Optional Fields	Description	Slot Type
scheduleType	Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the ActivatePipeline operation for each subsequent run. Values are: cron, ondemand, and timeseries.	Enumeration
securityGroupIds	The IDs of one or more Amazon EC2 Security Group Ids to use for the instances in the resource pool.	String
securityGroups	The IDs of one or more Amazon EC2 security group to use for the instances in the resource pool.	String
spotBidPrice	The Spot Instance bid price, in dollars. A decimal value between 0 and 20.00, exclusive.	String
subnetId	The ID of the Amazon EC2 subnet in which to start the instance.	String
terminateAfter	Terminate the resource after these many hours.	Period
useOnDemandOnLastAttempt	On the last attempt to request a Spot Instance, make a request for On-Demand Instances rather than a Spot Instance. This ensures that if all previous attempts have failed, the last attempt is not interrupted in the middle by changes in the Spot market.	Boolean
workerGroup	Field not allowed on this object	String

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String

Runtime Fields	Description	Slot Type
@cascadeFailedOn	Description of depedency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnableObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@failureReason	The reason for the resource failure.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

EmrCluster

Represents the configuration of an EMR cluster. This object is used by EmrActivity (p. 156) to launch a cluster.

Schedulers

Schedulers provide a way to specify resource allocation and job prioritization within a Hadoop cluster. Administrators or users can choose a scheduler for various classes of users and applications. A scheduler will possibly use queues to allocate resources to users and applications. You set up those queues when you create the cluster. You can then setup priority for certain types of work and user over others. This provides for efficient use of cluster resources, while allowing more than one user to submit work to the cluster. There are three types of scheduler available:

- FairScheduler — Attempts to schedule resources evenly over a significant period of time.
- CapacityScheduler — Uses queues to allow cluster administrators to assign users to queues of varying priority and resource allocation.
- Default — Used by the cluster, which could be configured by your site.

Amazon EMR 2.x, 3.x vs. 4.x platforms

AWS Data Pipeline supports EMR clusters based on release label emr-4.0.0 or later, which requires the use of the `releaseLabel` field for the corresponding `EmrCluster` object. For previous platforms known as AMI releases, use the `amiVersion` field instead. If you are using a self-managed `EmrCluster` object with a release label, use the most current Task Runner. For more information about TaskRunner, see Working with Task Runner (p. 263). You can configure all classifications found in the Amazon EMR configuration API. For a list of all configurations see the Configuring Applications topic in the Amazon EMR Release Guide: http://docs.aws.amazon.com/ElasticMapReduce/latest/ReleaseGuide/ as well as the the section called “EmrConfiguration” (p. 258) and the section called “Property” (p. 261) object references.

Examples

The following are examples of this object type.

Example 1: Launch an EMR cluster using the hadoopVersion field

The following example launches an EMR cluster using AMI version 1.0 and Hadoop 0.20.

```
{
  "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"]
}
```

Example 1: Launch an EMR cluster with release label emr-4.x or greater

The following example launches an EMR cluster using the newer `releaseLabel` field:

```
{
  "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"}
}
```

Example 2: Install additional software on your EMR cluster

`EmrCluster` provides the `supportedProducts` field that installs third-party software on an EMR cluster, for example installing a custom distribution of Hadoop like MapR. It accepts a comma-separated list of arguments for the third-party software to read and act on. The following example shows how to use the `supportedProducts` field of `EmrCluster` to create a custom MapR M3 edition cluster with Karmasphere Analytics installed, and run an `EmrActivity` object on it.

```
{
    "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"
}
```

Example 3: Disable server-side encryption on 3.x AMIs

An `EmrCluster` activity with a Hadoop version 2.x created by AWS Data Pipeline enables server-side encryption by default. If you would like to disable server-side encryption, you must specify a bootstrap action in the cluster object definition.

The following example creates an `EmrCluster` activity with server-side encryption disabled:

```
{
   "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"]
}
```

Example 3: Disable server-side encryption on 4.x releases

You must disable server-side encryption using a `EmrConfiguration` object.

The following example creates an `EmrCluster` activity with server-side encryption disabled:

```
   {
      "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"
    }
```

Example Configure Hadoop KMS Access Control Lists (ACLs) and Create Encryption Zones in HDFS

The following objects will create ACLs for Hadoop KMS and create encryption zones and corresponding encryption keys in HDFS:

```
{
      "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"
    }
```

Example 4: Specify custom IAM roles

By default, AWS Data Pipeline passes `DataPipelineDefaultRole` as the EMR service role and `DataPipelineDefaultResourceRole` as the EC2 instance profile to create resources on your behalf. However, you can create a custom EMR service role and a custom instance profile and use them instead. AWS Data Pipeline should have sufficient permissions to create clusters using the custom role, and you must add AWS Data Pipeline as a trusted entity.

The following example object specifies custom roles for the EMR cluster:

```
{
   "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"
}
```

Example Using EmrCluster Resource in AWS SDK for Java

The following shows how to use an EmrCluster and EmrActivity to create an Amazon EMR 4.x cluster to run a Spark step using the Java SDK:

```
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);

   }

}
```

When you create a custom IAM role, carefully consider the minimum permissions necessary for your cluster to perform its work. Be sure to grant access to required resources, such as files in Amazon S3 or data in Amazon RDS, Amazon Redshift or DynamoDB.

If you wish to set `visibleToAllUsers` to False, your role must have the proper permissions to do so. Note that `DataPipelineDefaultRole` does not have these permissions. You must either provide a union of the `DefaultDataPipelineResourceRole` and `DataPipelineDefaultRole` roles as the `EmrCluster` object role or create your own role for this purpose.

Syntax

Object Invocation Fields	Description	Slot Type
schedule	This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), users can create a parent object that has a schedule reference. For more information about example optional schedule configurations, see http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html	Reference Object, e.g. "schedule": {"ref":"myScheduleId"}

Optional Fields	Description	Slot Type
actionOnResourceFailure	The action taken after a resource failure for this resource. Valid values are "retryall", which retries all taks to the cluster for the specified duration, and "retrynone".	String
actionOnTaskFailure	The action taken after task failure for this resource. Valid values are "continue", meaning do not terminate the cluster, and "terminate."	String
additionalMasterSecurityGroupIds	The identifier of additional master security groups of the EMR cluster, which follows the form sg-01XXXX6a. For more information, see Amazon EMR Additional Security Groups in the Amazon Elastic MapReduce Developer Guide.	String
additionalSlaveSecurityGroupIds	The identifier of additional slave security groups of the EMR cluster, which follows the form sg-01XXXX6a.	String
amiVersion	The Amazon Machine Image (AMI) version that Amazon EMR uses to install the cluster nodes. For more information, see AMI Versions Supported in Amazon EMR in the Amazon Elastic MapReduce Developer Guide.	String
applications	applications to install in the cluster with comma separated arguments. By default, hive and pig will be installed.This parameter is applicable only for releaseLabel emr-4.0.0 and above	String
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
availabilityZone	The availability zone in which to run the cluster.	String
bootstrapAction	An action to run when the cluster starts. You can specify comma-separated arguments. To specify multiple actions, up to 255, add multiple bootstrapAction fields. The default behavior is to start the cluster without any bootstrap actions.	String
configuration	Configuration for the EMR cluster. This parameter is applicable only for releaseLabel emr-4.0.0 and above	Reference Object, e.g. "configuration": {"ref":"myEmrConfigurationId"}
coreInstanceBidPrice	The maximum dollar amount per hour for your Spot Instance bid. A decimal value between 0 and 20.00, exclusive. Setting this value enables Spot Instances for the EMR cluster core nodes. Must be used together with coreInstanceCount.	String
coreInstanceCount	The number of core nodes to use for the cluster.	Integer

Optional Fields	Description	Slot Type
coreInstanceType	The type of EC2 instance to use for core nodes. The default value is m1.small.	String
emrManagedMasterSecurityGroupId	The identifier of the master security group of the EMR cluster, which follows the form sg-01XXXX6a. For more information, see Configure Security Groups for Amazon EMR in the Amazon Elastic MapReduce Developer Guide.	String
emrManagedSlaveSecurityGroupId	The identifier of the slave security group of the EMR cluster, which follows the form sg-01XXXX6a.	String
enableDebugging	Enables debugging on the EMR cluster.	String
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
hadoopSchedulerType	The scheduler type of the cluster. Valid types are: PARALLEL_FAIR_SCHEDULING, PARALLEL_CAPACITY_SCHEDULING, and DEFAULT_SCHEDULER.	Enumeration
httpProxy	The proxy host that clients use to connect to AWS services.	Reference Object, e.g. "httpProxy": {"ref":"myHttpProxyId"}
initTimeout	The amount of time to wait for the resource to start.	Period
keyPair	The Amazon EC2 key pair to use to log onto the master node of the EMR cluster.	String
lateAfterTimeout	The elapsed time after pipeline start within which the object must start.	Period
masterInstanceBidPrice	The maximum dollar amount for your Spot Instance bid. A decimal value between 0 and 20.00, exclusive. Setting this value enables Spot Instances for the Amazon EMR cluster master node.	String
masterInstanceType	The type of EC2 instance to use for the master node. The default value is m1.small.	String
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref":"myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref":"myActionId"}

Optional Fields	Description	Slot Type
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref":"myActionId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
region	The code for the region that the EMR cluster should run in. By default, the cluster runs in the same region as the pipeline. You can run the cluster in the same region as a dependent data set.	Enumeration
releaseLabel	release label for the Amazon EMR cluster	String
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set then remote activities that do not report progres for the specified period may be considered stalled and so retried.	Period
resourceRole	The IAM role AWS Data Pipeline uses to create the EMR cluster. The default role is DataPipelineDefaultRole.	String
retryDelay	The timeout duration between two retry attempts.	Period
role	The IAM role passed to EMR to create EC2 nodes	String
runsOn	This Field is not allowed on this object.	Reference Object, e.g. "runsOn": {"ref":"myResourceId"}
scheduleType	Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the ActivatePipeline operation for each subsequent run. Values are: cron, ondemand, and timeseries.	Enumeration
subnetId	The identifier of the subnet to launch the cluster into.	String

Optional Fields	Description	Slot Type
supportedProducts	A parameter that installs third-party software on an EMR cluster, for example installing a third-party distribution of Hadoop.	String
taskInstanceBidPrice	The maximum dollar amount for your Spot Instance bid. A decimal value between 0 and 20.00, exclusive. Setting this value enables Spot Instances for the EMR cluster task nodes.	String
taskInstanceCount	The number of task nodes to use for the cluster.	Integer
taskInstanceType	The type of EC2 instance to use for task nodes.	String
terminateAfter	Terminate the resource after these many hours.	Period
useOnDemandOnLastAttempt	On the last attempt to request a resource, make a request for On-Demand Instances rather than Spot Instances. This ensures that if all previous attempts have failed, the last attempt is not interrupted in the middle by changes in the Spot market.	Boolean
workerGroup	Field not allowed on this object	String

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of depedency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnableObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@failureReason	The reason for the resource failure.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String

Runtime Fields	Description	Slot Type
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

See Also

- EmrActivity (p. 156)

HttpProxy

HttpProxy allows you to configure your own proxy and make Task Runner access the AWS Data Pipeline service through it. You do not need to configure a running Task Runner with this information.

Example

The following pipeline definition shows an HttpProxy object:

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

Syntax

Required Fields	Description	Slot Type
hostname	Host of the proxy which clients will use to connect to AWS Services	String
port	Port of the proxy host which the clients will use to connect to AWS Services	String

Optional Fields	Description	Slot Type
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
*password	Password for proxy	String
s3NoProxy	Disable the HTTP proxy when connecting to Amazon S3	Boolean
username	Username for proxy	String
windowsDomain	The Windows domain name for NTLM Proxy.	String
windowsWorkgroup	The Windows workgroup name for NTLM Proxy.	String

Runtime Fields	Description	Slot Type
@version	Pipeline version the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

Preconditions

The following are the AWS Data Pipeline precondition objects:

Objects

- DynamoDBDataExists (p. 223)

- DynamoDBTableExists (p. 225)
- Exists (p. 227)
- S3KeyExists (p. 230)
- S3PrefixNotEmpty (p. 232)
- ShellCommandPrecondition (p. 234)

DynamoDBDataExists

A precondition to check that data exists in a DynamoDB table.

Syntax

Required Fields	Description	Slot Type
role	Specifies the role to be used to execute the precondition.	String
tableName	DynamoDB Table to check.	String

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
lateAfterTimeout	The elapsed time after pipeline start within which the object must start.	Period
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref":"myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref":"myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref":"myActionId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
preconditionTimeout	The period from start after which precondition is marked as failed if still not satisfied	Period

Optional Fields	Description	Slot Type
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set then remote activities that do not report progres for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of depedency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnableObjectId"}
currentRetryCount	Number of times the precondition was tried in this attempt.	String
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
hostname	The host name of client that picked up the task attempt.	String
lastRetryTime	Last time when the precondition was tried within this attempt.	String
node	The node for which this precondition is being performed	Reference Object, e.g. "node": {"ref":"myRunnableObjectId"}
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String

Runtime Fields	Description	Slot Type
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

DynamoDBTableExists

A precondition to check that the DynamoDB table exists.

Syntax

Required Fields	Description	Slot Type
role	Specifies the role to be used to execute the precondition.	String
tableName	DynamoDB Table to check.	String

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
lateAfterTimeout	The elapsed time after pipeline start within which the object must start.	Period
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref":"myActionId"}

Optional Fields	Description	Slot Type
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref":"myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref":"myActionId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
preconditionTimeout	The period from start after which precondition is marked as failed if still not satisfied	Period
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set then remote activities that do not report progres for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of depedency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnableObjectId"}
currentRetryCount	Number of times the precondition was tried in this attempt.	String
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
hostname	The host name of client that picked up the task attempt.	String

Runtime Fields	Description	Slot Type
lastRetryTime	Last time when the precondition was tried within this attempt.	String
node	The node for which this precondition is being performed	Reference Object, e.g. "node": {"ref":"myRunnableObjectId"}
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

Exists

Checks whether a data node object exists.

Note
We recommend that you use system-managed preconditions instead. For more information, see Preconditions (p. 7).

Example

The following is an example of this object type. The `InputData` object references this object, `Ready`, plus another object that you'd define in the same pipeline definition file. `CopyPeriod` is a `Schedule` object.

```
{
  "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"
```

```
}
```

Syntax

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
lateAfterTimeout	The elapsed time after pipeline start within which the object must start.	Period
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref":"myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref":"myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref":"myActionId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
preconditionTimeout	The period from start after which precondition is marked as failed if still not satisfied	Period
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set then remote activities that do not report progres for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime

Runtime Fields	Description	Slot Type
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of depedency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnableObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
hostname	The host name of client that picked up the task attempt.	String
node	The node for which this precondition is being performed	Reference Object, e.g. "node": {"ref":"myRunnableObjectId"}
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

See Also

- ShellCommandPrecondition (p. 234)

S3KeyExists

Checks whether a key exists in an Amazon S3 data node.

Syntax

Required Fields	Description	Slot Type
role	Specifies the role to be used to execute the precondition.	String
s3Key	Amazon S3 key to check for existence.	String

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
lateAfterTimeout	The elapsed time after pipeline start within which the object must start.	Period
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref":"myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref":"myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref":"myActionId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
preconditionTimeout	The period from start after which precondition is marked as failed if still not satisfied	Period
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set then remote activities that do not report progres for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of depedency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnableObjectId"}
currentRetryCount	Number of times the precondition was tried in this attempt.	String
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
hostname	The host name of client that picked up the task attempt.	String
lastRetryTime	Last time when the precondition was tried within this attempt.	String
node	The node for which this precondition is being performed	Reference Object, e.g. "node": {"ref":"myRunnableObjectId"}
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

See Also

- ShellCommandPrecondition (p. 234)

S3PrefixNotEmpty

A precondition to check that the Amazon S3 objects with the given prefix (represented as a URI) are present.

Example

The following is an example of this object type using required, optional, and expression fields.

```
{
  "id" : "InputReady",
  "type" : "S3PrefixNotEmpty",
  "role" : "test-role",
  "s3Prefix" : "#{node.filePath}"
}
```

Syntax

Required Fields	Description	Slot Type
role	Specifies the role to be used to execute the precondition.	String
s3Prefix	Amazon S3 prefix to check for existence of objects.	String

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
lateAfterTimeout	The elapsed time after pipeline start within which the object must start.	Period

Optional Fields	Description	Slot Type
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref":"myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref":"myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref":"myActionId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
preconditionTimeout	The period from start after which precondition is marked as failed if still not satisfied	Period
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set then remote activities that do not report progres for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of depedency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnableObjectId"}
currentRetryCount	Number of times the precondition was tried in this attempt.	String
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String

Runtime Fields	Description	Slot Type
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
hostname	The host name of client that picked up the task attempt.	String
lastRetryTime	Last time when the precondition was tried within this attempt.	String
node	The node for which this precondition is being performed	Reference Object, e.g. "node": {"ref":"myRunnableObjectId"}
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

See Also

- ShellCommandPrecondition (p. 234)

ShellCommandPrecondition

A Unix/Linux shell command that can be run as a precondition.

Example

The following is an example of this object type.

```
{
  "id" : "VerifyDataReadiness",
  "type" : "ShellCommandPrecondition",
```

```
  "command" : "perl check-data-ready.pl"
}
```

Syntax

Required Group (One of the following is required)	Description	Slot Type
command	The command to run. This value and any associated parameters must function in the environment from which you are running the Task Runner.	String
scriptUri	An Amazon S3 URI path for a file to download and run as a shell command. Only one scriptUri or command field should be present. scriptUri cannot use parameters, use command instead.	String

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
lateAfterTimeout	The elapsed time after pipeline start within which the object must start.	Period
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref":"myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref":"myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref":"myActionId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
preconditionTimeout	The period from start after which precondition is marked as failed if still not satisfied	Period
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set then remote activities that do	Period

Optional Fields	Description	Slot Type
	not report progres for the specified period may be considered stalled and so retried.	
retryDelay	The timeout duration between two retry attempts.	Period
scriptArgument	Argument to be passed to shell script	String
stderr	The Amazon S3 path that receives redirected system error messages from the command. If you use the runsOn field, this must be an Amazon S3 path because of the transitory nature of the resource running your activity. However if you specify the workerGroup field, a local file path is permitted.	String
stdout	The Amazon S3 path that receives redirected output from the command. If you use the runsOn field, this must be an Amazon S3 path because of the transitory nature of the resource running your activity. However if you specify the workerGroup field, a local file path is permitted.	String

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of depedency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnableObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
hostname	The host name of client that picked up the task attempt.	String
node	The node for which this precondition is being performed	Reference Object, e.g. "node": {"ref":"myRunnableObjectId"}

Runtime Fields	Description	Slot Type
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

See Also

- ShellCommandActivity (p. 194)
- Exists (p. 227)

Databases

The following are the AWS Data Pipeline database objects:

Objects

- JdbcDatabase (p. 237)
- RdsDatabase (p. 239)
- RedshiftDatabase (p. 240)

JdbcDatabase

Defines a JDBC database.

Example

The following is an example of this object type.

```
{
```

```
  "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"
}
```

Syntax

Required Fields	Description	Slot Type
connectionString	The JDBC connection string to access the database.	String
jdbcDriverClass	The driver class to load before establishing the JDBC connection.	String
*password	The password to supply	String
username	The user name to supply when connecting to the database	String

Optional Fields	Description	Slot Type
databaseName	Name of the logical database to attach to	String
jdbcDriverJarUri	The location in Amazon S3 of the JDBC driver JAR file used to connect to the database. AWS Data Pipeline must have permission to read this JAR file.	String
jdbcProperties	Pairs of the form A=B that will be set as properties on jdbc connections for this database	String
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}

Runtime Fields	Description	Slot Type
@version	Pipeline version the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

RdsDatabase

Defines an Amazon RDS database.

Example

The following is an example of this object type.

```
{
  "id" : "MyRdsDatabase",
  "type" : "RdsDatabase",
  "region" : "us-east-1",
  "username" : "user_name",
  "*password" : "my_password",
  "rdsInstanceId" : "my_db_instance_identifier"
}
```

For the Oracle engine, the `jdbcDriverJarUri` field is required and you can specify the following driver: `http://www.oracle.com/technetwork/database/features/jdbc/jdbc-drivers-12c-download-1958347.html`. For the SQL Server engine, the `jdbcDriverJarUri` field is required and you can specify the following driver: `https://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=11774`. For the MySQL and PostgreSQL engines, the `jdbcDriverJarUri` field is optional.

Syntax

Required Fields	Description	Slot Type
*password	The password to supply	String
rdsInstanceId	The identifier of the DB instance.	String
username	The user name to supply when connecting to the database	String

Optional Fields	Description	Slot Type
databaseName	Name of the logical database to attach to	String
jdbcDriverJarUri	The location in Amazon S3 of the JDBC driver JAR file used to connect to the database. AWS Data Pipeline must have permission to read this JAR file. For the MySQL and PostgreSQL engines, the default driver is used if this field is not specified, but you can override the default using this field. For the Oracle and SQL Server engines, this field is required.	String
jdbcProperties	Pairs of the form A=B that will be set as properties on jdbc connections for this database	String
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}

Optional Fields	Description	Slot Type
region	The code for the region where the database exists. For example, us-east-1.	String

Runtime Fields	Description	Slot Type
@version	Pipeline version the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

RedshiftDatabase

Defines an Amazon Redshift database. `RedshiftDatabase` represents the properties of the database used by your pipeline.

Example

The following is an example of this object type.

```
{
  "id" : "MyRedshiftDatabase",
  "type" : "RedshiftDatabase",
  "clusterId" : "myRedshiftClusterId",
  "username" : "user_name",
  "*password" : "my_password",
  "databaseName" : "database_name"
}
```

By default, the object uses the Postgres driver, which requires the `clusterId` field. To use the Redshift driver, specify the Redshift database connection string from the Amazon Redshift console (starts with "jdbc:redshift:") in the `connectionString` field instead.

Syntax

Required Fields	Description	Slot Type
*password	The password to supply	String
username	The user name to supply when connecting to the database	String

Required Group (One of the following is required)	Description	Slot Type
clusterId	The identifier provided by the user when the Amazon Redshift cluster was created. For example, if the endpoint for your Amazon Redshift cluster is mydb.example.us-east-1.redshift.amazonaws.com, the correct identifier is mydb. In the Amazon Redshift console, you can get this value from Cluster Identifier or Cluster Name.	String
connectionString	The JDBC endpoint for connecting to an Amazon Redshift instance owned by an account different than the pipeline. Note that you can't specify both connectionString and clusterId.	String

Optional Fields	Description	Slot Type
databaseName	Name of the logical database to attach to	String
jdbcProperties	Pairs of the form A=B that will be set as properties on jdbc connections for this database	String
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
region	The code for the region where the database exists. For example, us-east-1.	Enumeration

Runtime Fields	Description	Slot Type
@version	Pipeline version the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

Data Formats

The following are the AWS Data Pipeline data format objects:

Objects

CSV Data Format

A comma-delimited data format where the column separator is a comma and the record separator is a newline character.

Example

The following is an example of this object type.

```
{
  "id" : "MyOutputDataType",
  "type" : "CSV",
  "column" : [
    "Name STRING",
    "Score INT",
    "DateOfBirth TIMESTAMP"
  ]
}
```

Syntax

Optional Fields	Description	Slot Type
column	Column name with datatype specified by each field for the data described by this data node. Ex: hostname STRING For multiple values, use column names and data types separated by a space.	String
escapeChar	A character, for example "\", that instructs the parser to ignore the next character.	String
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}

Runtime Fields	Description	Slot Type
@version	Pipeline version the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String

System Fields	Description	Slot Type
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

Custom Data Format

A custom data format defined by a combination of a certain column separator, record separator, and escape character.

Example

The following is an example of this object type.

```
{
  "id" : "MyOutputDataType",
  "type" : "Custom",
  "columnSeparator" : ",",
  "recordSeparator" : "\n",
  "column" : [
    "Name STRING",
    "Score INT",
    "DateOfBirth TIMESTAMP"
  ]
}
```

Syntax

Required Fields	Description	Slot Type
columnSeparator	A character that indicates the end of a column in a data file.	String

Optional Fields	Description	Slot Type
column	Column name with datatype specified by each field for the data described by this data node. Ex: hostname STRING For multiple values, use column names and data types separated by a space.	String
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
recordSeparator	A character that indicates the end of a row in a data file, for example "\n". Only single characters are supported.	String

Runtime Fields	Description	Slot Type
@version	Pipeline version the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

DynamoDBDataFormat

Applies a schema to a DynamoDB table to make it accessible by a Hive query. `DynamoDBDataFormat` is used with a `HiveActivity` object and a `DynamoDBDataNode` input and output. `DynamoDBDataFormat` requires that you specify all columns in your Hive query. For more flexibility to specify certain columns in a Hive query or Amazon S3 support, see DynamoDBExportDataFormat (p. 246).

Note
DynamoDB Boolean types are not mapped to Hive Boolean types. However, it is possible to map DynamoDB integer values of 0 or 1 to Hive Boolean types.

Example

The following example shows how to use `DynamoDBDataFormat` to assign a schema to a `DynamoDBDataNode` input, which allows a `HiveActivity` object to access the data by named columns and copy the data to a `DynamoDBDataNode` output.

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

Syntax

Optional Fields	Description	Slot Type
column	Column name with datatype specified by each field for the data described by this data node. Ex: hostname STRING For multiple values, use column names and data types separated by a space.	String
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}

Runtime Fields	Description	Slot Type
@version	Pipeline version the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

DynamoDBExportDataFormat

Applies a schema to an DynamoDB table to make it accessible by a Hive query. Use `DynamoDBExportDataFormat` with a `HiveCopyActivity` object and `DynamoDBDataNode` or `S3DataNode` input and output. `DynamoDBExportDataFormat` has the following benefits:

- Provides both DynamoDB and Amazon S3 support
- Allows you to filter data by certain columns in your Hive query
- Exports all attributes from DynamoDB even if you have a sparse schema

Note
DynamoDB Boolean types are not mapped to Hive Boolean types. However, it is possible to map DynamoDB integer values of 0 or 1 to Hive Boolean types.

Example

The following example shows how to use `HiveCopyActivity` and `DynamoDBExportDataFormat` to copy data from one `DynamoDBDataNode` to another, while filtering based on a time stamp.

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

Syntax

Optional Fields	Description	Slot Type
column	Column name with datatype specified by each field for the data described by this data node. Ex: hostname STRING	String
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}

Runtime Fields	Description	Slot Type
@version	Pipeline version the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

RegEx Data Format

A custom data format defined by a regular expression.

Example

The following is an example of this object type.

```
{
  "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"
  ]
}
```

Syntax

Optional Fields	Description	Slot Type
column	Column name with datatype specified by each field for the data described by this data node. Ex: hostname STRING For multiple values, use column names and data types separated by a space.	String
inputRegEx	The regular expression to parse an S3 input file. inputRegEx provides a way to retrieve columns from relatively unstructured data in a file.	String
outputFormat	The column fields retrieved by inputRegEx, but referenced as %1$s %2$s using Java formatter syntax.	String
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}

Runtime Fields	Description	Slot Type
@version	Pipeline version the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

TSV Data Format

A comma-delimited data format where the column separator is a tab character and the record separator is a newline character.

Example

The following is an example of this object type.

```
{
  "id" : "MyOutputDataType",
  "type" : "TSV",
  "column" : [
    "Name STRING",
    "Score INT",
    "DateOfBirth TIMESTAMP"
  ]
}
```

Syntax

Optional Fields	Description	Slot Type
column	Column name and datatype for the data described by this data node. For example `"Name STRING"` denotes a column named `Name` with fields of datatype `STRING`. Separate multiple column name and datatype pairs with commas (as shown in the example).	String
columnSeparator	The character that separates fields in one column from fields in the next column. Defaults to '\t'.	String
escapeChar	A character, for example "\", that instructs the parser to ignore the next character.	String
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
recordSeparator	The character that separates records. Defaults to '\n'.	String

Runtime Fields	Description	Slot Type
@version	Pipeline version the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String

System Fields	Description	Slot Type
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

Actions

The following are the AWS Data Pipeline action objects:

Objects

- SnsAlarm (p. 250)
- Terminate (p. 251)

SnsAlarm

Sends an Amazon SNS notification message when an activity fails or finishes successfully.

Example

The following is an example of this object type. The values for `node.input` and `node.output` come from the data node or activity that references this object in its `onSuccess` field.

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

Syntax

Required Fields	Description	Slot Type
message	The body text of the Amazon SNS notification.	String
role	The IAM role to use to create the Amazon SNS alarm.	String
subject	The subject line of the Amazon SNS notification message.	String
topicArn	The destination Amazon SNS topic ARN for the message.	String

Optional Fields	Description	Slot Type
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}

Runtime Fields	Description	Slot Type
node	The node for which this action is being performed	Reference Object, e.g. "node": {"ref":"myRunnableObjectId"}
@version	Pipeline version the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

Terminate

An action to trigger the cancellation of a pending or unfinished activity, resource, or data node. AWS Data Pipeline attempts to put the activity, resource, or data node into the CANCELLED state if it does not start by the `lateAfterTimeout` value.

Example

The following is an example of this object type. In this example, the `onLateAction` field of `MyActivity` contains a reference to the action `DefaultAction1`. When you provide an action for `onLateAction`, you must also provide a `lateAfterTimeout` value to indicate the period of time since the scheduled start of the pipeline after which the activity is considered late.

```
{
  "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"
}
```

Syntax

Optional Fields	Description	Slot Type
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}

Runtime Fields	Description	Slot Type
node	The node for which this action is being performed	Reference Object, e.g. "node": {"ref":"myRunnableObjectId"}
@version	Pipeline version the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

Schedule

Defines the timing of a scheduled event, such as when an activity runs.

Note
When a schedule's start time is in the past, AWS Data Pipeline backfills your pipeline and begins scheduling runs immediately beginning at the specified start time. For testing/development, use a relatively short interval. Otherwise, AWS Data Pipeline attempts to queue and schedule all runs of your pipeline for that interval. AWS Data Pipeline attempts to prevent accidental backfills if the pipeline component `scheduledStartTime` is earlier than 1 day ago by blocking pipeline activation.

Examples

The following is an example of this object type. It defines a schedule of every hour starting at 00:00:00 hours on 2012-09-01 and ending at 00:00:00 hours on 2012-10-01. The first period ends at 01:00:00 on 2012-09-01.

```
{
  "id" : "Hourly",
  "type" : "Schedule",
  "period" : "1 hours",
  "startDateTime" : "2012-09-01T00:00:00",
  "endDateTime" : "2012-10-01T00:00:00"
}
```

The following pipeline will start at the `FIRST_ACTIVATION_DATE_TIME` and run every hour until 22:00:00 hours on 2014-04-25.

```
{
     "id": "SchedulePeriod",
     "name": "SchedulePeriod",
     "startAt": "FIRST_ACTIVATION_DATE_TIME",
     "period": "1 hours",
     "type": "Schedule",
     "endDateTime": "2014-04-25T22:00:00"
   }
```

The following pipeline will start at the `FIRST_ACTIVATION_DATE_TIME` and run every hour and complete after three occurrences.

```
{
     "id": "SchedulePeriod",
     "name": "SchedulePeriod",
     "startAt": "FIRST_ACTIVATION_DATE_TIME",
     "period": "1 hours",
     "type": "Schedule",
     "occurrences": "3"
   }
```

The following pipeline will start at 22:00:00 on 2014-04-25, run hourly, and end after three occurrences.

```
{
     "id": "SchedulePeriod",
     "name": "SchedulePeriod",
     "startDateTime": "2014-04-25T22:00:00",
     "period": "1 hours",
     "type": "Schedule",
     "occurrences": "3"
   }
```

On-demand using the Default object

```
{
  "name": "Default",
  "resourceRole": "DataPipelineDefaultResourceRole",
  "role": "DataPipelineDefaultRole",
  "scheduleType": "ondemand"
}
```

The following examples demonstrate how a Schedule can be inherited from the default object, be explicitly set for that object, or be given by a parent reference:

Schedule inherited from Default object

```
{
  "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!'"
    }
  ]
}
```

Explicit schedule on the object

```
{
  "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!'"
    }
  ]
}
```

Schedule from Parent reference

```
{
```

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

Syntax

Required Fields	Description	Slot Type
period	How often the pipeline should run. The format is "N [minutes|hours|days|weeks|months]", where N is a number followed by one of the time specifiers. For example, "15 minutes", runs the pipeline every 15 minutes. The minimum period is 15 minutes and the maximum period is 3 years.	Period

Required Group (One of the following is required)	Description	Slot Type
startAt	The date and time at which to start the scheduled pipeline runs. Valid value is FIRST_ACTIVATION_DATE_TIME, which is deprecated in favor of creating an on-demand pipeline.	Enumeration
startDateTime	The date and time to start the scheduled runs. You must use either startDateTime or startAt but not both.	DateTime

Optional Fields	Description	Slot Type
endDateTime	The date and time to end the scheduled runs. Must be a date and time later than the value of startDateTime or startAt. The default behavior is to schedule runs until the pipeline is shut down.	DateTime
occurrences	The number of times to execute the pipeline after it's activated. You can't use occurrences with endDateTime.	Integer
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}

Runtime Fields	Description	Slot Type
@version	Pipeline version the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@firstActivationTime	The time of object creation.	DateTime
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

Utilities

The following utility objects configure other pipeline objects:

Topics

ShellScriptConfig

Use with an Activity to run a shell script for preActivityTaskConfig and postActivityTaskConfig. This object is available for HadoopActivity (p. 162), HiveActivity (p. 168), HiveCopyActivity (p. 174), and PigActivity (p. 179). You specify an S3 URI and a list of arguments for the script.

Example

A ShellScriptConfig with arguments:

```
{
   "id" : "ShellScriptConfig_1”,
   "name" : “prescript”,
   "type" : "ShellScriptConfig",
   "scriptUri": “s3://my-bucket/shell-cleanup.sh”,
   "scriptArgument" : ["arg1","arg2"]
 }
```

Syntax

This object includes the following fields.

Optional Fields	Description	Slot Type
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
scriptArgument	A list of argument(s) to use with the shell script.	String
scriptUri	The script URI in Amazon S3 that should be downloaded and run.	String

Runtime Fields	Description	Slot Type
@version	Pipeline version the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

EmrConfiguration

The EmrConfiguration object is the configuration used for EMR clusters with releases 4.0.0 or greater. Configurations (as a list) is a parameter to the RunJobFlow API call. The configuration API for Amazon EMR takes a classification and properties. AWS Data Pipeline uses EmrConfiguration with corresponding Property objects to configure an EmrCluster (p. 210) application such as Hadoop, Hive, Spark, or Pig on EMR clusters launched in a pipeline execution. Because configuration can only be changed for new clusters, you cannot provide a EmrConfiguration object for existing resources. For more information, see http://docs.aws.amazon.com/ElasticMapReduce/latest/ReleaseGuide/.

Example

The following configuration object sets the `io.file.buffer.size` and `fs.s3.block.size` properties in `core-site.xml`:

```
[
   {
      "classification":"core-site",
      "properties":
      {
         "io.file.buffer.size": "4096",
         "fs.s3.block.size": "67108864"
      }
   }
]
```

The corresponding pipeline object definition uses a EmrConfiguration object and a list of Property objects in the `property` field:

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

The following example is a nested configuration used to set the Hadoop environment with the `hadoop-env` classfication:

```
[
  {
    "classification": "hadoop-env",
    "properties": {},
    "configurations": [
      {
        "classification": "export",
        "properties": {
          "YARN_PROXYSERVER_HEAPSIZE": "2396"
        }
      }
    ]
  }
]
```

The corresponding pipeline definition object that uses this configuration is below:

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

Syntax

This object includes the following fields.

Required Fields	Description	Slot Type
classification	classification for the configuration	String

Optional Fields	Description	Slot Type
configuration	sub-configuration for this configuration	Reference Object, e.g. "configuration": {"ref":"myEmrConfigurationId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}
property	configuration property	Reference Object, e.g. "property": {"ref":"myPropertyId"}

Runtime Fields	Description	Slot Type
@version	Pipeline version the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

See Also

- EmrCluster (p. 210)
- Property (p. 261)
- Amazon EMR Release Guide

Property

A single key-value property for use with an EmrConfiguration object.

Example

The following pipeline definition shows an EmrConfiguration object and corresponding Property objects to launch an EmrCluster:

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

Syntax

This object includes the following fields.

Required Fields	Description	Slot Type
key	key	String
value	value	String

Optional Fields	Description	Slot Type
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref":"myBaseObjectId"}

Runtime Fields	Description	Slot Type
@version	Pipeline version the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

See Also

- EmrCluster (p. 210)
- EmrConfiguration (p. 258)
- Amazon EMR Release Guide

Working with Task Runner

Task Runner is a task agent application that polls AWS Data Pipeline for scheduled tasks and executes them on Amazon EC2 instances, Amazon EMR clusters, or other computational resources, reporting status as it does so. Depending on your application, you may choose to:

- Allow AWS Data Pipeline to install and manage one or more Task Runner applications for you. When a pipeline is activated, the default `Ec2Instance` or `EmrCluster` object referenced by an activity runsOn field is automatically created. AWS Data Pipeline takes care of installing Task Runner on an EC2 instance or on the master node of an EMR cluster. In this pattern, AWS Data Pipeline can do most of the instance or cluster management for you.
- Run all or parts of a pipeline on resources that you manage. The potential resources include a long-running Amazon EC2 instance, an Amazon EMR cluster, or a physical server. You can install a task runner (which can be either Task Runner or a custom task agent of your own devise) almost anywhere, provided that it can communicate with the AWS Data Pipeline web service. In this pattern, you assume almost complete control over which resources are used and how they are managed, and you must manually install and configure Task Runner. To do so, use the procedures in this section, as described in Executing Work on Existing Resources Using Task Runner (p. 264).

Task Runner on AWS Data Pipeline-Managed Resources

When a resource is launched and managed by AWS Data Pipeline, the web service automatically installs Task Runner on that resource to process tasks in the pipeline. You specify a computational resource (either an Amazon EC2 instance or an Amazon EMR cluster) for the `runsOn` field of an activity object. When AWS Data Pipeline launches this resource, it installs Task Runner on that resource and configures it to process all activity objects that have their `runsOn` field set to that resource. When AWS Data Pipeline terminates the resource, the Task Runner logs are published to an Amazon S3 location before it shuts down.

For example, if you use the `EmrActivity` in a pipeline, and specify an `EmrCluster` resource in the `runsOn` field. When AWS Data Pipeline processes that activity, it launches an Amazon EMR cluster and installs Task Runner onto the master node. This Task Runner then processes the tasks for activities that have their `runsOn` field set to that `EmrCluster` object. The following excerpt from a pipeline definition shows this relationship between the two objects.

```
{
  "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"
}
```

If you have multiple AWS Data Pipeline-managed resources in a pipeline, Task Runner is installed on each of them, and they all poll AWS Data Pipeline for tasks to process.

Executing Work on Existing Resources Using Task Runner

You can install Task Runner on computational resources that you manage, such as an Amazon EC2 instance, or a physical server or workstation. Task Runner can be installed anywhere, on any compatible hardware or operating system, provided that it can communicate with the AWS Data Pipeline web service.

This approach can be useful when, for example, you want to use AWS Data Pipeline to process data that is stored inside your organization's firewall. By installing Task Runner on a server in the local network, you can access the local database securely and then poll AWS Data Pipeline for the next task to run. When AWS Data Pipeline ends processing or deletes the pipeline, the Task Runner instance remains running on your computational resource until you manually shut it down. The Task Runner logs persist after pipeline execution is complete.

To use Task Runner on a resource that you manage, you must first download Task Runner, and then install it on your computational resource, using the procedures in this section.

Note

You can only install Task Runner on Linux, UNIX, or Mac OS. Task Runner is not supported on the Windows operating system.

To connect a Task Runner that you've installed to the pipeline activities it should process, add a `workerGroup` field to the object, and configure Task Runner to poll for that worker group value. You do this by passing the worker group string as a parameter (for example, `--workerGroup=wg-12345`) when you run the Task Runner JAR file.

```
{
  "id" : "CreateDirectory",
  "type" : "ShellCommandActivity",
  "workerGroup" : "wg-12345",
  "command" : "mkdir new-directory"
}
```

Installing Task Runner

This section explains how to install and configure Task Runner and its prerequisites. Installation is a straightforward manual process.

To install Task Runner

1. Task Runner requires Java version 1.6 or later. To determine whether Java is installed, and the version that is running, use the following command:

   ```
   java -version
   ```

   If you do not have Java 1.6 or later installed on your computer, you can download the latest version from http://www.oracle.com/technetwork/java/index.html. Download and install Java, and then proceed to the next step.
2. Download `TaskRunner-1.0.jar` from https://s3.amazonaws.com/datapipeline-us-east-1/us-east-1/software/latest/TaskRunner/TaskRunner-1.0.jar and then copy it into a folder on the target computing resource. For Amazon EMR clusters running `EmrActivity` tasks, you will install Task Runner on the master node of the cluster. Additionally, download `mysql-connector-java-bin.jar` from http://s3.amazonaws.com/datapipeline-prod-us-east-1/software/latest/TaskRunner/mysql-connector-java-bin.jar and copy it into the same folder where you install Task Runner.
3. Task Runner needs to connect to the AWS Data Pipeline web service to process your commands. In this step, you will configure Task Runner with an AWS account that has permissions to create or manage data pipelines.

   Create a JSON file named `credentials.json` (you can use a different name if you prefer), which specifies an access key ID and secret access key using the format `{ "accessKeyId": MyAccessKeyID, "secretAccessKey": MySecretAccessKey }` . Copy the file to the directory where you installed Task Runner.

   For CLI access, you need an access key ID and secret access key. For more information about creating access keys, see How Do I Get Security Credentials? in the *AWS General Reference*.
4. Task Runner connects to the AWS Data Pipeline web service using HTTPS. If you are using an AWS resource, ensure that HTTPS is enabled in the appropriate routing table and subnet ACL. If you are using a firewall or proxy, ensure that port 443 is open.

(Optional) Granting Task Runner Access to Amazon RDS

Amazon RDS allows you to control access to your DB instances using database security groups (DB security groups). A DB security group acts like a firewall controlling network access to your DB instance. By default, network access is turned off for your DB instances. You must modify your DB security groups to let Task Runner access your Amazon RDS instances. Task Runner gains Amazon RDS access from the instance on which it runs, so the accounts and security groups that you add to your Amazon RDS instance depend on where you install Task Runner.

To grant access to Task Runner in EC2-Classic

1. Open the Amazon RDS console.
2. In the navigation pane, select **Instances**, and then select your DB instance.
3. Under **Security and Network**, click the security group, which opens the **Security Groups** page with this DB security group selected. Click the details icon for the DB security group.

4. Under **Security Group Details**, create a rule with the appropriate **Connection Type** and **Details**. These fields depend on where Task Runner is running, as described here:

   - `Ec2Resource`
     - **Connection Type**: `EC2 Security Group`
       Details: *`my-security-group-name`* (the name of the security group you created for the EC2 instance)
   - `EmrResource`
     - **Connection Type**: `EC2 Security Group`
       Details: `ElasticMapReduce-master`
     - **Connection Type**: `EC2 Security Group`
       Details: `ElasticMapReduce-slave`
   - Your local environment (on-premises)
     - **Connection Type**: `CIDR/IP`:
       Details: *`my-ip-address`* (the IP address of your computer or the IP address range of your network, if your computer is behind a firewall)

5. Click Add.

To grant access to Task Runner in EC2-VPC

1. Open the Amazon RDS console.
2. In the navigation pane, click **Instances**.
3. Click the details icon for the DB instance. Under **Security and Network**, click the link to the security group, which takes you to the Amazon EC2 console. If you're using the old console design for security groups, switch to the new console design by clicking the icon that's displayed at the top of the console page.
4. From the **Inbound** tab, click **Edit** and then click **Add Rule**. Specify the database port that you used when you launched the DB instance. The source depends on where Task Runner is running, as described here:

   - `Ec2Resource`
     - *`my-security-group-id`* (the ID of the security group you created for the EC2 instance)
   - `EmrResource`
     - *`master-security-group-id`* (the ID of the `ElasticMapReduce-master` security group)
     - *`slave-security-group-id`* (the ID of the `ElasticMapReduce-slave` security group)
   - Your local environment (on-premises)
     - *`ip-address`* (the IP address of your computer or the IP address range of your network, if your computer is behind a firewall)

5. Click **Save**.

Starting Task Runner

In a new command prompt window that is set to the directory where you installed Task Runner, start Task Runner with the following command.

```
java -jar TaskRunner-1.0.jar --config ~/credentials.json --workerGroup=myWorkerGroup --
region=MyRegion --logUri=s3://mybucket/foldername
```

The `--config` option points to your credentials file.

The `--workerGroup` option specifies the name of your worker group, which must be the same value as specified in your pipeline for tasks to be processed.

The `--region` option specifies the service region from which to pull tasks to execute.

The `--logUri` option is used for pushing your compressed logs to a location in Amazon S3.

When Task Runner is active, it prints the path to where log files are written in the terminal window. The following is an example.

```
Logging to /Computer_Name/.../output/logs
```

Task Runner should be run detached from your login shell. If you are using a terminal application to connect to your computer, you may need to use a utility like nohup or screen to prevent the Task Runner application from exiting when you log out. For more information about command line options, see Task Runner Configuration Options (p. 267).

Verifying Task Runner Logging

The easiest way to verify that Task Runner is working is to check whether it is writing log files. Task Runner writes hourly log files to the directory, `output/logs`, under the directory where Task Runner is installed. The file name is `Task Runner.log.YYYY-MM-DD-HH`, where HH runs from 00 to 23, in UDT. To save storage space, any log files older than eight hours are compressed with GZip.

Task Runner Threads and Preconditions

Task Runner uses a thread pool for each of tasks, activities, and preconditions. The default setting for `--tasks` is 2, meaning that there will be two threads allocated from the tasks pool and each thread will poll the AWS Data Pipeline service for new tasks. Thus, `--tasks` is a performance tuning attribute that can be used to help optimize pipeline throughput.

Pipeline retry logic for preconditions happens in Task Runner. Two precondition threads are allocated to poll AWS Data Pipeline for precondition objects. Task Runner honors the precondition object **retryDelay** and **preconditionTimeout** fields that you define on preconditions.

In many cases, decreasing the precondition polling timeout and number of retries can help to improve the performance of your application. Similarly, applications with long-running preconditions may need to have the timeout and retry values increased. For more information about precondition objects, see Preconditions (p. 7).

Task Runner Configuration Options

These are the configuration options available from the command line when you launch Task Runner.

Command Line Parameter	Description
`--help`	Command line help. Example: `Java -jar TaskRunner-1.0.jar --help`
`--config`	The path and file name of your `credentials.json` file.
`--accessId`	Your AWS access key ID for Task Runner to use when making requests.

Command Line Parameter	Description
	The `--accessID` and `--secretKey` options provide an alternative to using a credentials.json file. If a `credentials.json` is also provided, the `--accessID` and `--secretKey` options will take precedence.
`--secretKey`	Your AWS secret key for Task Runner to use when making requests. For more information, see `--accessID`.
`--endpoint`	An endpoint is a URL that is the entry point for a web service. The AWS Data Pipeline service endpoint in the region where you are making requests. Optional. In general, it is sufficient to specify a region, and you do not need to set the endpoint. For a listing of AWS Data Pipeline regions and endpoints, see AWS Data Pipeline Regions and Endpoints in the *AWS General Reference*.
`--workerGroup`	The name of the worker group that Task Runner will retrieve work for. Required. When Task Runner polls the web service, it uses the credentials you supplied and the value of `workerGroup` to select which (if any) tasks to retrieve. You can use any name that is meaningful to you; the only requirement is that the string must match between the Task Runner and its corresponding pipeline activities. The worker group name is bound to a region. Even if there are identical worker group names in other regions, Task Runner will always get tasks from the region specified in `--region`.
`--taskrunnerId`	The ID of the task runner to use when reporting progress. Optional.
`--output`	The Task Runner directory for log output files. Optional. Log files are stored in a local directory until they are pushed to Amazon S3. This option will override the default directory.
`--tasks`	The number of task poll threads to run simultaneously. Optional. The default is 2.
`--region`	The region to use. Optional, but it is recommended to always set the region. If you do not specify the region, Task Runner retrieves tasks from the default service region, `us-east-1`. Other supported regions are: `eu-west-1`, `ap-northeast-1`, `ap-southeast-2`, `us-west-2`.

Command Line Parameter	Description
`--logUri`	The Amazon S3 destination path for Task Runner to back up log files to every hour. When Task Runner terminates, active logs in the local directory will be pushed to the Amazon S3 destination folder.
`--proxyHost`	The host of the proxy which Task Runner clients will use to connect to AWS Services.
`--proxyPort`	Port of the proxy host which the Task Runner clients will use to connect to AWS Services.
`--proxyUsername`	The username for proxy.
`--proxyPassword`	The password for proxy.
`--proxyDomain`	The Windows domain name for NTLM Proxy.
`--proxyWorkstation`	The Windows workstation name for NTLM Proxy.

Using Task Runner with a Proxy

If you are using a proxy host, you can either specify its configuration when invoking Task Runner or set the environment variable, HTTPS_PROXY. The environment variable used with Task Runner will accept the same configuration used for the AWS Command Line Interface.

Task Runner and Custom AMIs

When you specify an `Ec2Resource` object for your pipeline, AWS Data Pipeline creates an EC2 instance for you, using an AMI that installs and configures Task Runner for you. Note that a PV-compatible instance type is required in this case. Alternatively, you can create a custom AMI with Task Runner, and then specify the ID of this AMI using the `imageId` field of the `Ec2Resource` object. For more information, see Ec2Resource (p. 204).

A custom AMI must meet the following requirements for AWS Data Pipeline to use it successfully for Task Runner:

- Create the AMI in the same region that the instances will run in. For more information, see Creating Your Own AMI in the *Amazon EC2 User Guide for Linux Instances*.
- Ensure that the virtualization type of the AMI is supported by the instance type you plan to use. For example, the I2 and G2 instance types require an HVM AMI and the T1, C1, M1, and M2 instance types require a PV AMI. For more information, see Linux AMI Virtualization Types in the *Amazon EC2 User Guide for Linux Instances*.
- Install the following software:
  - Linux
  - Bash
  - wget
  - unzip
  - Java 1.6 or newer
  - cloud-init
- Create and configure a user account named `ec2-user`.

Troubleshooting

When you have a problem with AWS Data Pipeline, the most common symptom is that a pipeline won't run. You can use the data that the console and CLI provide to identity the problem and find a solution.

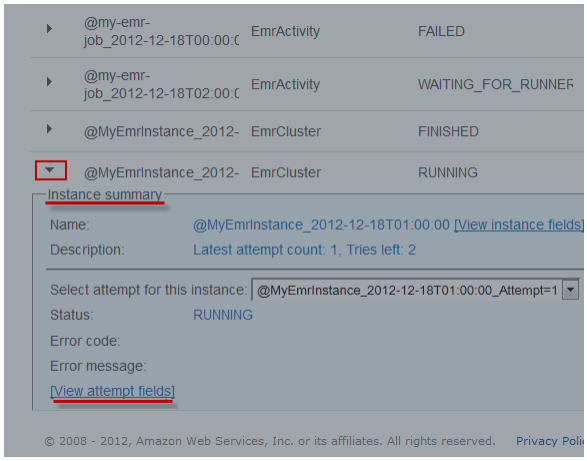
Contents

Locating Errors in Pipelines

The AWS Data Pipeline console is a convenient tool to visually monitor the status of your pipelines and easily locate any errors related to failed or incomplete pipeline runs.

To locate errors about failed or incomplete runs with the console

1. On the **List Pipelines** page, if the **Status** column of any of your pipeline instances shows a status other than **FINISHED**, either your pipeline is waiting for some precondition to be met or it has failed and you need to troubleshoot the pipeline.
2. On the **List Pipelines** page, in the **Details** column of your pipeline, click **View instance details**.
3. Click the triangle next to an instance; the **Instance summary** panel opens to show the details of the selected instance.
4. Click **View instance fields** to see additional details of the instance. If the status of your selected instance is **FAILED**, the details box has an entry indicating the reason for failure. For example, `@failureReason = Resource not healthy terminated`.
5. In the **Instance summary** pane, in the **Select attempt for this instance** field, select the attempt number.
6. In the **Instance summary** pane, click **View attempt fields** to see details of fields associated with the selected attempt.



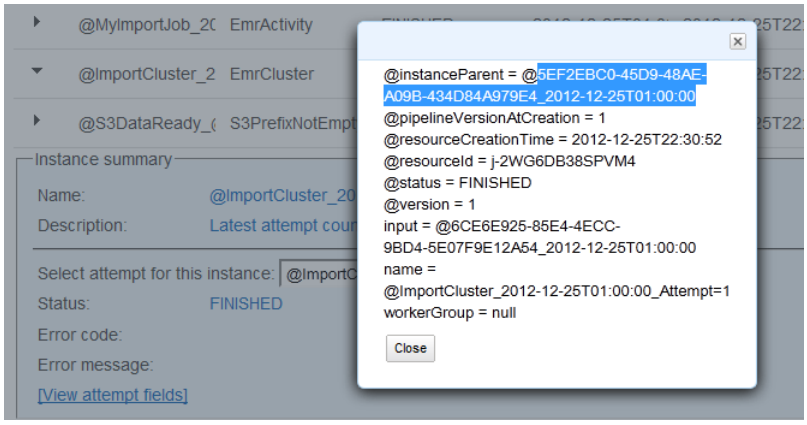
7. To take an action on your incomplete or failed instance, select an action (`Rerun|Cancel|Mark Finished`) from the **Action** column of the instance.

Identifying the Amazon EMR Cluster that Serves Your Pipeline

If an `EMRCluster` or `EMRActivity` fails and the error information provided by the AWS Data Pipeline console is unclear, you can identify the Amazon EMR cluster that serves your pipeline using the Amazon EMR console. This helps you locate the logs that Amazon EMR provides to get more details about errors that occur.

To see more detailed Amazon EMR error information,

1. In the AWS Data Pipeline console, on the **Instance details:** screen, select the **EmrCluster**, click **View attempt fields**, and copy the **instanceParent** value from the attempt fields dialog as shown in the example below.



2. Navigate to the Amazon EMR console and search for a cluster with the matching **instanceParent** value in its name and click **Debug**.

   Note
   For the **Debug** button to function, your pipeline definition must have set the EmrActivity `enableDebugging` option to `true` and the `EmrLogUri` option to a valid path.

3. Now that you know which Amazon EMR cluster contains the error that causes your pipeline failure, follow the Troubleshooting Tips in the *Amazon EMR Developer Guide*.

Interpreting Pipeline Status Details

The various status levels displayed in the AWS Data Pipeline console and CLI indicate the condition of a pipeline and its components. The pipeline status is simply an overview of a pipeline; to see more information, view the status of individual pipeline components. You can do this by clicking through a pipeline in the console or retrieving pipeline component details using the CLI.

Status Codes

`ACTIVATING`

The component or resource is being started, such as an EC2 instance.

`CANCELED`

The component was canceled by a user or AWS Data Pipeline before it could run. This can happen automatically when a failure occurs in a different component or resource that this component depends on.

`CASCADE_FAILED`

The component or resource was canceled as a result of a cascade failure from one of its dependencies, but the component was probably not the original source of the failure.

`DEACTIVATING`

The pipeline is being deactivated.

`FAILED`

The component or resource encountered an error and stopped working. When a component or resource fails, it can cause cancelations and failures to cascade to other components that depend on it.

`FINISHED`

The component completed its assigned work.

`INACTIVE`

The pipeline was deactivated.

`PAUSED`

The component was paused and is not currently performing its work.

`PENDING`

The pipeline is ready to be activated for the first time.

`RUNNING`

The resource is running and ready to receive work.

`SHUTTING_DOWN`

The resource is shutting down after successfully completing its work.

`SKIPPED`

The component skipped intervals of execution after the pipeline was activated using a timestamp that is later than the current schedule.

`TIMEDOUT`

The resource exceeded the `terminateAfter` threshold and was stopped by AWS Data Pipeline. After the resource reaches this status, AWS Data Pipeline ignores the `actionOnResourceFailure`, `retryDelay`, and `retryTimeout` values for that resource. This status applies only to resources.

`VALIDATING`

The pipeline definition is being validated by AWS Data Pipeline.

`WAITING_FOR_RUNNER`

The component is waiting for its worker client to retrieve a work item. The component and worker client relationship is controlled by the `runsOn` or `workerGroup` fields defined by that component.

`WAITING_ON_DEPENDENCIES`

The component is verifying that its default and user-configured preconditions are met before performing its work.

Locating Error Logs

This section explains how to find the various logs that AWS Data Pipeline writes, which you can use to determine the source of certain failures and errors.

Pipeline Logs

We recommend that you configure pipelines to create log files in a persistent location, such as in the following example where you use the `pipelineLogUri` field on a pipeline's `Default` object to cause all pipeline components to use an Amazon S3 log location by default (you can override this by configuring a log location in a specific pipeline component).

Note
Task Runner stores its logs in a different location by default, which may be unavailable when the pipeline finishes and the instance that runs Task Runner terminates. For more information, see Verifying Task Runner Logging (p. 267).

To configure the log location using the AWS Data Pipeline CLI in a pipeline JSON file, begin your pipeline file with the following text:

```
{ "objects": [
{
  "id":"Default",
  "pipelineLogUri":"s3://mys3bucket/error_logs"
},
...
```

After you configure a pipeline log directory, Task Runner creates a copy of the logs in your directory, with the same formatting and file names described in the previous section about Task Runner logs.

Hadoop Job and Amazon EMR Step Logs

With any Hadoop-based activity such as HadoopActivity (p. 162), HiveActivity (p. 168), or PigActivity (p. 179) you can view Hadoop job logs at the location returned in the runtime slot, hadoopJobLog. EmrActivity (p. 156) has its own logging features and those logs are stored using the location chosen by Amazon EMR and returned by the runtime slot, emrStepLog. For more information, see View Log Files in the Amazon EMR Developer Guide.

Resolving Common Problems

This topic provides various symptoms of AWS Data Pipeline problems and the recommended steps to solve them.

Contents

Pipeline Stuck in Pending Status

A pipeline that appears stuck in the PENDING status indicates that a pipeline has not yet been activated, or activation failed due to an error in the pipeline definition. Ensure that you did not receive any errors when you submitted your pipeline using the AWS Data Pipeline CLI or when you attempted to save or activate your pipeline using the AWS Data Pipeline console. Additionally, check that your pipeline has a valid definition.

To view your pipeline definition on the screen using the CLI:

```
datapipeline --get --id df-EXAMPLE_PIPELINE_ID
```

Ensure that the pipeline definition is complete, check your closing braces, verify required commas, check for missing references, and other syntax errors. It is best to use a text editor that can visually validate the syntax of JSON files.

Pipeline Component Stuck in Waiting for Runner Status

If your pipeline is in the SCHEDULED state and one or more tasks appear stuck in the WAITING_FOR_RUNNER state, ensure that you set a valid value for either the runsOn or workerGroup fields for those tasks. If both values are empty or missing, the task cannot start because there is no association between the task and a worker to perform the tasks. In this situation, you've defined work but haven't defined what computer will do that work. If applicable, verify that the workerGroup value assigned to the pipeline component is exactly the same name and case as the workerGroup value that you configured for Task Runner.

> **Note**
> If you provide a runsOn value and workerGroup exists, workerGroup is ignored.

Another potential cause of this problem is that the endpoint and access key provided to Task Runner is not the same as the AWS Data Pipeline console or the computer where the AWS Data Pipeline CLI tools are installed. You might have created new pipelines with no visible errors, but Task Runner polls the wrong location due to the difference in credentials, or polls the correct location with insufficient permissions to identify and run the work specified by the pipeline definition.

Pipeline Component Stuck in WAITING_ON_DEPENDENCIES Status

If your pipeline is in the SCHEDULED state and one or more tasks appear stuck in the WAITING_ON_DEPENDENCIES state, make sure your pipeline's initial preconditions have been met. If the preconditions of the first object in the logic chain are not met, none of the objects that depend on that first object will be able to move out of the WAITING_ON_DEPENDENCIES state.

For example, consider the following excerpt from a pipeline definition. In this case, the InputData object has a precondition 'Ready' specifying that the data must exist before the InputData object is complete. If the data does not exist, the InputData object remains in the WAITING_ON_DEPENDENCIES state, waiting for the data specified by the path field to become available. Any objects that depend on InputData likewise remain in a WAITING_ON_DEPENDENCIES state waiting for the InputData object to reach the FINISHED state.

```
{
    "id": "InputData",
    "type": "S3DataNode",
    "filePath": "s3://elasticmapreduce/samples/wordcount/wordSplitter.py",
    "schedule":{"ref":"MySchedule"},
    "precondition": "Ready"
},
{
    "id": "Ready",
    "type": "Exists"
...
```

Also, check that your objects have the proper permissions to access the data. In the preceding example, if the information in the credentials field did not have permissions to access the data specified in the path field, the InputData object would get stuck in a WAITING_ON_DEPENDENCIES state because it cannot access the data specified by the path field, even if that data exists.

It is also possible that a resource communicating with Amazon S3 does not have a public IP address associated with it. For example, an Ec2Resource in a public subnet must have a public IP address associated with it.

Lastly, under certain conditions, resource instances can reach the WAITING_ON_DEPENDENCIES state much earlier than their associated activities are scheduled to start, which may give the impression that the resource or the activity is failing. For more information about the behavior of resources and the schedule type setting, see the *Resources Ignore Schedule Type* section in the Scheduling Pipelines (p. 15) topic.

Run Doesn't Start When Scheduled

Check that you chose the correct schedule type that determines whether your task starts at the beginning of the schedule interval (Cron Style Schedule Type) or at the end of the schedule interval (Time Series Schedule Type).

Additionally, check that you have properly specified the dates in your schedule objects and that the startDateTime and endDateTime values are in UTC format, such as in the following example:

```
{
    "id": "MySchedule",
    "startDateTime": "2012-11-12T19:30:00",
    "endDateTime":"2012-11-12T20:30:00",
    "period": "1 Hour",
    "type": "Schedule"
},
```

Pipeline Components Run in Wrong Order

You might notice that the start and end times for your pipeline components are running in the wrong order, or in a different sequence than you expect. It is important to understand that pipeline components can start running simultaneously if their preconditions are met at start-up time. In other words, pipeline components do not execute sequentially by default; if you need a specific execution order, you must control the execution order with preconditions and dependsOn fields. Verify that you are using the dependsOn field populated with a reference to the correct prerequisite pipeline components, and that all the necessary pointers between components are present to achieve the order you require.

EMR Cluster Fails With Error: The security token included in the request is invalid

Verify your IAM roles, policies, and trust relationships as described in IAM Roles for AWS Data Pipeline (p. 66).

Insufficient Permissions to Access Resources

Permissions that you set on IAM roles determine whether AWS Data Pipeline can access your EMR clusters and EC2 instances to run your pipelines. Additionally, IAM provides the concept of trust relationships that go further to allow creation of resources on your behalf. For example, when you create a pipeline that uses an EC2 instance to run a command to move data, AWS Data Pipeline can provision this EC2 instance for you. If you encounter problems, especially those involving resources that you can access manually but AWS Data Pipeline cannot, verify your IAM roles, policies, and trust relationships as described in IAM Roles for AWS Data Pipeline (p. 66).

Status Code: 400 Error Code: PipelineNotFoundException

This error means that your IAM default roles might not have the required permissions necessary for AWS Data Pipeline to function correctly. For more information, see IAM Roles for AWS Data Pipeline (p. 66).

Creating a Pipeline Causes a Security Token Error

You receive the following error when you try to create a pipeline:

Failed to create pipeline with 'pipeline_name'. Error: UnrecognizedClientException - The security token included in the request is invalid.

Cannot See Pipeline Details in the Console

The AWS Data Pipeline console pipeline filter applies to the *scheduled* start date for a pipeline, without regard to when the pipeline was submitted. It is possible to submit a new pipeline using a scheduled start date that occurs in the past, which the default date filter may not show. To see the pipeline details, change your date filter to ensure that the scheduled pipeline start date fits within the date range filter.

Error in remote runner Status Code: 404, AWS Service: Amazon S3

This error means that Task Runner could not access your files in Amazon S3. Verify that:

- You have credentials correctly set
- The Amazon S3 bucket that you are trying to access exists
- You are authorized to access the Amazon S3 bucket

Access Denied - Not Authorized to Perform Function datapipeline:

In the Task Runner logs, you may see an error that is similar to the following:

- ERROR Status Code: 403
- AWS Service: DataPipeline
- AWS Error Code: AccessDenied
- AWS Error Message: User: arn:aws:sts::XXXXXXXXXXXX:federated-user/i-XXXXXXXX is not authorized to perform: datapipeline:PollForTask.

> **Note**
> In the this error message, PollForTask may be replaced with names of other AWS Data Pipeline permissions.

This error message indicates that the IAM role you specified needs additional permissions necessary to interact with AWS Data Pipeline. Ensure that your IAM role policy contains the following lines, where PollForTask is replaced with the name of the permission you want to add (use * to grant all permissions). For more information about how to create a new IAM role and apply a policy to it, see Managing IAM Policies in the *Using IAM* guide.

```
{
"Action": [ "datapipeline:PollForTask" ],
"Effect": "Allow",
"Resource": ["*"]
}
```

Older Amazon EMR AMIs May Create False Data for Large CSV Files

On EMR AMIs previous to 3.9 (3.8 and below) AWS Data Pipeline uses a custom InputFormat to read and write CSV files for use with MapReduce jobs. This is used when the service stages tables to and from Amazon S3. An issue with this InputFormat was discovered where reading records from large CSV files may result in producing tables that are not correctly copied. This issue was fixed in later Amazon EMR releases. Please use Amazon EMR AMI 3.9 or an Amazon EMR release 4.0.0 or greater.

Increasing AWS Data Pipeline Limits

Occasionally, you may exceed specific AWS Data Pipeline system limits. For example, the default pipeline limit is 20 pipelines with 50 objects in each. If you discover that you need more pipelines than the limit, consider merging multiple pipelines to create fewer pipelines with more objects in each. For more information about the AWS Data Pipeline limits, see AWS Data Pipeline Limits (p. 280). However, if you are unable to work around the limits using the pipeline merge technique, request an increase in your capacity using this form: Data Pipeline Limit Increase.

Logging AWS Data Pipeline API Calls By Using AWS CloudTrail

AWS Data Pipeline is integrated with CloudTrail, a service that captures API calls made by or on behalf of AWS Data Pipeline in your AWS account and delivers the log files to an Amazon S3 bucket that you specify. CloudTrail captures API calls from the AWS Data Pipeline console or from the AWS Data Pipeline API. Using the information collected by CloudTrail, you can determine what request was made to AWS Data Pipeline, the source IP address from which the request was made, who made the request, when it was made, and so on. For more information about CloudTrail, including how to configure and enable it, see the AWS CloudTrail User Guide.

AWS Data Pipeline Information in CloudTrail

When CloudTrail logging is enabled in your AWS account, API calls made to AWS Data Pipeline actions are tracked in log files. AWS Data Pipeline records are written together with other AWS service records in a log file. CloudTrail determines when to create and write to a new file based on a time period and file size.

All of the AWS Data Pipeline actions are logged and are documented in the AWS Data Pipeline API Reference Actions chapter. For example, calls to the **CreatePipeline** action generate entries in the CloudTrail log files.

Every log entry contains information about who generated the request. The user identity information in the log helps you determine whether the request was made with root or IAM user credentials, with temporary security credentials for a role or federated user, or by another AWS service. For more information, see the **userIdentity** field in the CloudTrail Event Reference.

You can store your log files in your bucket for as long as you want, but you can also define Amazon S3 lifecycle rules to archive or delete log files automatically. By default, your log files are encrypted by using Amazon S3 server-side encryption (SSE).

You can choose to have CloudTrail publish Amazon SNS notifications when new log files are delivered if you want to take quick action upon log file delivery. For more information, see Configuring Amazon SNS Notifications.

You can also aggregate AWS Data Pipeline log files from multiple AWS regions and multiple AWS accounts into a single Amazon S3 bucket. For more information, see Aggregating CloudTrail Log Files to a Single Amazon S3 Bucket.

Understanding AWS Data Pipeline Log File Entries

CloudTrail log files can contain one or more log entries where each entry is made up of multiple JSON-formatted events. A log entry represents a single request from any source and includes information about the requested operation, any parameters, the date and time of the action, and so on. The log entries are not guaranteed to be in any particular order. That is, they are not an ordered stack trace of the public API calls.

The following example shows a CloudTrail log entry that demonstrates the `CreatePipeline` operation:

```
    {
  "Records": [
    {
      "eventVersion": "1.02",
      "userIdentity": {
        "type": "Root",
        "principalId": "123456789012",
        "arn": "arn:aws:iam::user-account-id:root",
        "accountId": "user-account-id",
        "accessKeyId": "user-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": "user-account-id"
    },
      ...additional entries
  ]
}
```

AWS Data Pipeline Limits

To ensure there is capacity for all users, AWS Data Pipeline imposes limits on the resources that you can allocate and the rate at which you can allocate resources.

Contents

Account Limits

The following limits apply to a single AWS account. If you require additional capacity, you can use the Amazon Web Services Support Center request form to increase your capacity.

Attribute	Limit	Adjustable
Number of pipelines	100	Yes
Number of objects per pipeline	100	Yes
Number of active instances per object	5	Yes
Number of fields per object	50	No
Number of UTF8 bytes per field name or identifier	256	No
Number of UTF8 bytes per field	10,240	No
Number of UTF8 bytes per object	15,360 (including field names)	No
Rate of creation of a instance from an object	1 per 5 minutes	No

Attribute	Limit	Adjustable
Retries of a pipeline activity	5 per task	No
Minimum delay between retry attempts	2 minutes	No
Minimum scheduling interval	15 minutes	No
Maximum number of roll-ups into a single object	32	No
Maximum number of EC2 instances per Ec2Resource object	1	No

Web Service Call Limits

AWS Data Pipeline limits the rate at which you can call the web service API. These limits also apply to AWS Data Pipeline agents that call the web service API on your behalf, such as the console, CLI, and Task Runner.

The following limits apply to a single AWS account. This means the total usage on the account, including that by IAM users, cannot exceed these limits.

The burst rate lets you save up web service calls during periods of inactivity and expend them all in a short amount of time. For example, CreatePipeline has a regular rate of 1 call each 5 seconds. If you don't call the service for 30 seconds, you will have 6 calls saved up. You could then call the web service 6 times in a second. Because this is below the burst limit and keeps your average calls at the regular rate limit, your calls are not throttled.

If you exceed the rate limit and the burst limit, your web service call fails and returns a throttling exception. The default implementation of a worker, Task Runner, automatically retries API calls that fail with a throttling exception, with a back off so that subsequent attempts to call the API occur at increasingly longer intervals. If you write a worker, we recommend that you implement similar retry logic.

These limits are applied against an individual AWS account.

API	Regular rate limit	Burst limit
ActivatePipeline	1 call per second	100 calls
CreatePipeline	1 call per second	100 calls
DeletePipeline	1 call per second	100 calls
DescribeObjects	2 calls per second	100 calls
DescribePipelines	1 call per second	100 calls
GetPipelineDefinition	1 call per second	100 calls
PollForTask	2 calls per second	100 calls
ListPipelines	1 call per second	100 calls

API	Regular rate limit	Burst limit
PutPipelineDefinition	1 call per second	100 calls
QueryObjects	2 calls per second	100 calls
ReportTaskProgress	10 calls per second	100 calls
SetTaskStatus	10 calls per second	100 calls
SetStatus	1 call per second	100 calls
ReportTaskRunnerHeartbeat	1 call per second	100 calls
ValidatePipelineDefinition	1 call per second	100 calls

Scaling Considerations

AWS Data Pipeline scales to accommodate a huge number of concurrent tasks and you can configure it to automatically create the resources necessary to handle large workloads. These automatically-created resources are under your control and count against your AWS account resource limits. For example, if you configure AWS Data Pipeline to automatically create a 20-node Amazon EMR cluster to process data and your AWS account has an EC2 instance limit set to 20, you may inadvertently exhaust your available backfill resources. As a result, consider these resource restrictions in your design or increase your account limits accordingly.

If you require additional capacity, you can use the Amazon Web Services Support Center request form to increase your capacity.

AWS Data Pipeline Resources

The following are resources to help you use AWS Data Pipeline.

- **AWS Data Pipeline Product Information**–The primary web page for information about AWS Data Pipeline.
- **AWS Data Pipeline Technical FAQ** – Covers the top 20 questions developers ask about this product.
- **Release Notes** – Provide a high-level overview of the current release. They specifically note any new features, corrections, and known issues.
- **AWS Data Pipeline Discussion Forums** – A community-based forum for developers to discuss technical questions related to Amazon Web Services.

- **Classes & Workshops** – Links to role-based and specialty courses as well as self-paced labs to help sharpen your AWS skills and gain practical experience.
- **AWS Developer Tools** – Links to developer tools, SDKs, IDE toolkits, and command line tools for developing and managing AWS applications.
- **AWS Whitepapers** – Links to a comprehensive list of technical AWS whitepapers, covering topics such as architecture, security, and economics and authored by AWS Solutions Architects or other technical experts.
- **AWS Support Center** – The hub for creating and managing your AWS Support cases. Also includes links to other helpful resources, such as forums, technical FAQs, service health status, and AWS Trusted Advisor.
- **AWS Support** – The primary web page for information about AWS Support, a one-on-one, fast-response support channel to help you build and run applications in the cloud.
- **Contact Us** – A central contact point for inquiries concerning AWS billing, account, events, abuse, and other issues.
- **AWS Site Terms** – Detailed information about our copyright and trademark; your account, license, and site access; and other topics.

Document History

This documentation is associated with the 2012-10-29 version of AWS Data Pipeline.

Latest documentation update: 22 February 2016.

Change	Description	Release Date
Add support for On-demand pipelines	• Added support for On-demand pipelines, which allows you to re-run a pipeline by activating it again. For more information, see On-demand (p. 16).	22 February 2016
Additional support for RDS databases	• Added `rdsInstanceId`, `region`, and `jdbcDriverJarUri` to RdsDatabase (p. 239). • Updated `database` in SqlActivity (p. 199) to also support `RdsDatabase`.	17 August 2015
Additional JDBC support	• Updated `database` in SqlActivity (p. 199) to also support `JdbcDatabase`. • Added `jdbcDriverJarUri` to JdbcDatabase (p. 237) • Added `initTimeout` to Ec2Resource (p. 204) and EmrCluster (p. 210). • Added `runAsUser` to Ec2Resource (p. 204).	7 July 2015
HadoopActivity, Availability Zone, and Spot Support	• Added support for submitting parallel work to Hadoop clusters. For more information, see HadoopActivity (p. 162). • Added the ability to request Spot Instances with Ec2Resource (p. 204) and EmrCluster (p. 210). • Added the ability to launch `EmrCluster` resources in a specified availability zone.	1 June 2015
Deactivating pipelines	Added support for deactivating active pipelines. For more information, see Deactivating Your Pipeline (p. 43).	7 April 2015
Updated templates and console	Added new templates as reflected in the console. Updated the Getting Started chapter to use the **Getting Started with ShellCommandActivity** template. For more information, see Creating Pipelines Using Console Templates (p. 18).	25 November 2014

Change	Description	Release Date
VPC support	Added support for launching resources into a virtual private cloud (VPC). For more information, see Launching Resources for Your Pipeline into a VPC (p. 48).	12 March 2014
Region support	Added support for multiple service regions. In addition to `us-east-1`, AWS Data Pipeline is supported in `eu-west-1`, `ap-northeast-1`, `ap-southeast-2`, and `us-west-2`.	20 February 2014
Redshift support	Added support for Redshift in AWS Data Pipeline, including a new console template (Copy to Redshift) and a tutorial to demonstrate the template. For more information, see Copy Data to Amazon Redshift Using AWS Data Pipeline (p. 107), RedshiftDataNode (p. 139), RedshiftDatabase (p. 240), and RedshiftCopyActivity (p. 187).	6 November 2013
PigActivity	Added PigActivity, which provides native support for Pig. For more information, see PigActivity (p. 179).	15 October 2013
New console template, activity, and data format	Added the new CrossRegion DynamoDB Copy console template, including the new HiveCopyActivity and DynamoDBExportDataFormat.	21 August 2013
Cascading failures and reruns	Added information about AWS Data Pipeline cascading failure and rerun behavior. For more information, see Cascading Failures and Reruns (p. 53).	8 August 2013
Troubleshooting video	Added the AWS Data Pipeline Basic Troubleshooting video. For more information, see Troubleshooting (p. 270).	17 July 2013
Editing active pipelines	Added more information about editing active pipelines and rerunning pipeline components. For more information, see Editing Your Pipeline (p. 40).	17 July 2013
Use resources in different regions	Added more information about using resources in different regions. For more information, see Using a Pipeline with Resources in Multiple Regions (p. 51).	17 June 2013
WAITING_ON_DEPENDENCIES status	CHECKING_PRECONDITIONS status changed to WAITING_ON_DEPENDENCIES and added the @waitingOn runtime field for pipeline objects.	20 May 2013
DynamoDBDataFormat	Added DynamoDBDataFormat template.	23 April 2013
Process Web Logs video and Spot Instances support	Introduced the video "Process Web Logs with AWS Data Pipeline, Amazon EMR, and Hive", and Amazon EC2 Spot Instances support.	21 February 2013
	The initial release of the AWS Data Pipeline Developer Guide.	20 December 2012