# Table of Contents

What is AWS Data Pipeline? ................................................................. 1
How Does AWS Data Pipeline Work? .................................................. 1
Setting Up .......................................................................................... 3
  Signing Up for AWS ........................................................................... 3
  (Optional) Installing a CLI .............................................................. 3
  (Optional) Granting Access to Resources ........................................ 4
  Setting Up IAM Roles ....................................................................... 4
  Granting IAM Users Access to the Console ..................................... 6
Getting Started with AWS Data Pipeline ............................................ 7
  Create the Pipeline ........................................................................... 8
  Monitor the Running Pipeline .......................................................... 8
  View the Output .............................................................................. 9
  Delete the Pipeline ......................................................................... 9
Working with Pipelines ....................................................................... 10
Data Pipeline Concepts ...................................................................... 10
  Pipeline Definition ......................................................................... 11
  Pipeline Components, Instances, and Attempts ............................... 12
  Lifecycle of a Pipeline .................................................................... 13
  Lifecycle of a Pipeline Task ............................................................ 13
  Task Runners .................................................................................. 14
  Data Nodes .................................................................................... 14
  Databases ....................................................................................... 15
  Activities ....................................................................................... 15
  Preconditions ................................................................................ 15
  Resources ...................................................................................... 16
  Actions .......................................................................................... 18
  Roles and Permissions ................................................................... 18
Scheduling Pipelines ........................................................................ 18
  Creating a Schedule Using the Console ........................................... 19
  Time Series Style vs. Cron Style .................................................... 19
  Backfill Tasks ................................................................................ 20
  Maximum Resource Efficiency Using Schedules ............................. 20
  Time Zones .................................................................................. 21
  Protecting Against Overwriting Data .............................................. 21
Creating Pipelines .......................................................................... 21
  Creating Pipelines Using Console Templates ................................. 21
  Creating Pipelines Using the Console Manually .............................. 30
Viewing Your Pipelines ................................................................. 33
  Interpreting Schedule Status Codes .............................................. 35
  Interpreting Pipeline and Component Health State ....................... 36
  Viewing Your Pipeline Definitions ............................................... 36
  Viewing Pipeline Instance Details ................................................ 38
  Viewing Pipeline Logs .................................................................. 39
Editing Your Pipelines ..................................................................... 40
Cloning Your Pipelines .................................................................... 42
Deleting Your Pipelines .................................................................... 42
Staging Data and Tables with Activities ......................................... 43
  Data Staging with ShellCommandActivity .................................... 43
  Table Staging with Hive and Staging-supported Data Nodes ............ 44
  Table Staging with Hive and Staging-unsupported Data Nodes ....... 45
Launching Resources into a VPC .................................................... 46
  Create and Configure a VPC .......................................................... 47
  Set Up Connectivity Between Resources ....................................... 47
  Configure the Resource ............................................................... 49
Using Spot Instances in a Pipeline ................................................... 50
<table>
<thead>
<tr>
<th>Action</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install the CLI</td>
<td>295</td>
</tr>
<tr>
<td></td>
<td>295</td>
</tr>
<tr>
<td>Install Ruby</td>
<td>296</td>
</tr>
<tr>
<td>Install RubyGems</td>
<td>297</td>
</tr>
<tr>
<td>Install the Required Ruby Gems</td>
<td>297</td>
</tr>
<tr>
<td>Install the CLI</td>
<td>298</td>
</tr>
<tr>
<td>Configure Credentials for the CLI</td>
<td>298</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLI Reference</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Options</td>
<td>301</td>
</tr>
<tr>
<td>Command Line Syntax</td>
<td>301</td>
</tr>
<tr>
<td>Examples</td>
<td>301</td>
</tr>
<tr>
<td>Related Commands</td>
<td>301</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Options</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>300</td>
</tr>
<tr>
<td>Syntax</td>
<td>300</td>
</tr>
<tr>
<td>Options</td>
<td>301</td>
</tr>
<tr>
<td>Common Options</td>
<td>301</td>
</tr>
<tr>
<td>Output</td>
<td>301</td>
</tr>
<tr>
<td>Examples</td>
<td>301</td>
</tr>
<tr>
<td>Related Commands</td>
<td>301</td>
</tr>
</tbody>
</table>

| Description         | 302  |
| Syntax              | 302  |
| Options             | 302  |
| Common Options      | 302  |
| Output              | 302  |
| Examples            | 302  |
| Related Commands    | 302  |

| Description         | 303  |
| Syntax              | 303  |
| Options             | 303  |
| Common Options      | 303  |
| Output              | 303  |
| Examples            | 303  |
| Related Commands    | 303  |

| Description         | 304  |
| Syntax              | 304  |
| Options             | 304  |
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.

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 Elastic MapReduce (Amazon EMR) cluster over those logs to generate traffic reports.

In this example, AWS Data Pipeline would schedule the daily tasks to copy data and the weekly task to launch the Amazon EMR cluster. AWS Data Pipeline would also ensure that Amazon EMR waits for the final day's data to be uploaded to Amazon S3 before it began its analysis, even if there is an unforeseen delay in uploading the logs.

AWS Data Pipeline handles the ambiguities of real-world data management. You define the parameters of your data transformations and AWS Data Pipeline enforces the logic that you've set up.

How Does AWS Data Pipeline Work?

Three main 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. 53)*.

• The **AWS Data Pipeline web service** interprets the pipeline definition and assigns tasks to workers to move and transform data.

• **Task Runner** polls the AWS Data Pipeline web service for tasks and then performs those tasks. In the previous example, Task Runner would 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. 14)*.

The following illustration below shows how these components work together. If the pipeline definition supports non-serialized tasks, AWS Data Pipeline can manage tasks for multiple task runners working in parallel.
Setting Up AWS Data Pipeline

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

Tasks

• Signing Up for AWS (p. 3)
• (Optional) Installing a Command Line Interface (p. 3)
• (Optional) Granting Access to Resources (p. 4)

After you complete these tasks, you can start using AWS Data Pipeline. For a basic tutorial, see Process Access Logs Using Amazon EMR with Hive (p. 59).

Signing 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

2. Follow the on-screen instructions.

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

(Optional) Installing a Command Line Interface

If you prefer to use a command line interface to automate the process of creating and managing pipelines, you can install and use the AWS Command Line Interface (CLI) which provides commands for a broad set of AWS products, and is supported on Windows, Mac, and Linux. To get started, see the AWS Command Line Interface User Guide.
(Optional) Granting Access to Resources

Your security credentials identify you to services in AWS and grant you unlimited use of your AWS resources. 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 pipeline’s resources.

Contents

• Setting Up IAM Roles (p. 4)
• Granting IAM Users Access to the Console (p. 6)

Setting Up IAM Roles

AWS Data Pipeline requires IAM roles to determine what actions your pipelines can perform and who can access your pipeline’s resources. The AWS Data Pipeline console creates the following roles for you:

• DataPipelineDefaultRole
• DataPipelineDefaultResourceRole

If you are using a CLI or an API, you must create these IAM roles, apply policies to them, and update the trusted entities list to include these roles.

To set up the required IAM roles for a CLI or API

1. Create DataPipelineDefaultRole and apply the following policy. For more information, see Managing IAM Policies in the Using IAM guide.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```
2. Create DataPipelineDefaultResourceRole and apply the following policy.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:List*",
        "s3:Put*",
        "s3:Get*",
        "s3:DeleteObject",
        "dynamodb:DescribeTable",
        "dynamodb:Scan",
        "dynamodb:Query",
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:UpdateTable",
        "rds:DescribeDBInstances",
        "rds:DescribeDBSecurityGroups",
        "redshift:DescribeClusters",
        "redshift:DescribeClusterSecurityGroups",
        "cloudwatch:PutMetricData",
        "datapipeline:*"
      ],
      "Resource": ["*"
    ]
  ]
}
```

3. Define a trusted entities list, which indicates the entities or services that have permission to use your roles. You can use the following IAM trust relationship definition to allow AWS Data Pipeline and Amazon EC2 to use your roles. For more information, see Modifying a Role in the Using IAM guide.
Granting IAM Users Access to the Console

Your AWS account has the necessary permissions to use the AWS Data Pipeline console. However, when you add IAM users to your account, you must use the following minimum IAM policy to grant them access to the AWS Data Pipeline console. For more information, see Managing IAM Policies in the Using IAM guide.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:*",
        "datapipeline:*",
        "dynamodb:DescribeTable",
        "iam:AddRoleToInstanceProfile",
        "iam:CreateInstanceProfile",
        "iam:CreateRole",
        "iam:GetInstanceProfile",
        "iam:GetRole",
        "iam:ListInstanceProfiles",
        "iam:ListInstanceProfilesForRole",
        "iam:ListRoles",
        "iam:PassRole",
        "iam:PutRolePolicy",
        "rds:DescribeDBInstances",
        "rds:DescribeDBSecurityGroups",
        "redshift:DescribeClusters",
        "redshift:DescribeClusterSecurityGroups",
        "s3:List*",
        "sns:ListTopics"
      ],
      "Resource": "*
    }
  ]
}
```
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. 15) that define the work to perform, data nodes (p. 14) that define the location and type of input and output data, and a schedule (p. 18) that determines when the activities are performed.

Pipeline Objects

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. The pipeline uses the following objects:

ShellCommandActivity (p. 233)
- Reads the input log file and counts the number of errors.

S3DataNode (p. 187) (input)
- The S3 bucket that contains the input log file.

S3DataNode (p. 187) (output)
- The S3 bucket for the output.

Ec2Resource (p. 244)
- 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. 292)
- Defines that the activity is performed every 15 minutes for an hour.

Prerequisites

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

Tasks
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.
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 this region.
   a. If you haven't created a pipeline in this region, the console displays an introductory screen. Click *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. Click *Create new pipeline*.
4. In *Name*, enter a name.
5. (Optional) In *Description*, enter a description.
6. Select *Build using a template*, and then select *Getting Started using ShellCommandActivity*.
7. Under *Parameters*, leave *S3 input location* and *Shell command to run* with their default values. Click the folder icon next to *S3 output location*, select one of your buckets or folders, and then click *Select*.
8. Under *Schedule*, leave the default values. The pipeline runs will start when you activate the pipeline, and run 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. Click the folder icon under *S3 location for logs*, select one of your buckets or folders, and then click *Select*.
10. Under *Security/Access*, leave *IAM roles* set to *Default*.
11. Click *Activate*.
   If you prefer, you can click *Edit in Architect* to modify this pipeline. For example, you can add Amazon SNS notifications or 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 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, 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. 156).

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 using the console

1. In the List Pipelines page, click the check box next to your pipeline.
2. Click Delete.
3. In the confirmation dialog box, click Delete to confirm the delete request.

If you are finished with the output from this tutorial, delete the output folders from your Amazon S3 bucket.
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 AWS Data Pipeline (p. 3).

**Contents**
- Data Pipeline Concepts (p. 10)
- Scheduling Pipelines (p. 18)
- Creating Pipelines (p. 21)
- Viewing Your Pipelines (p. 33)
- Editing Your Pipelines (p. 40)
- Cloning Your Pipelines (p. 42)
- Deleting Your Pipelines (p. 42)
- Staging Data and Tables with Pipeline Activities (p. 43)
- Launching Resources for Your Pipeline into a VPC (p. 46)
- Using Amazon EC2 Spot Instances in a Pipeline (p. 50)
- Using a Pipeline with Resources in Multiple Regions (p. 50)
- Cascading Failures and Reruns (p. 51)
- Pipeline Definition File Syntax (p. 53)
- Working with the API (p. 55)

**Data Pipeline Concepts**

The following sections describe the concepts and components in AWS Data Pipeline:

**Topics**
- Pipeline Definition (p. 11)
- Pipeline Components, Instances, and Attempts (p. 12)
- Lifecycle of a Pipeline (p. 13)
- Lifecycle of a Pipeline Task (p. 13)
- Task Runners (p. 14)
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:

Data Nodes (p. 14)
- The location of input data for a task or the location where output data is to be stored.

Activities (p. 15)
- A definition of work to perform on a schedule using a computational resource and typically input and output data nodes.

Preconditions (p. 15)
- A conditional statement that must be true before an action can run.

Scheduling Pipelines (p. 18)
- Defines the timing of a scheduled event, such as when an activity runs.

Resources (p. 16)
- The computational resource that performs the work that a pipeline defines.
Actions (p. 18)

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. 53).

**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 retries and related settings.
Lifecycle of a Pipeline

The pipeline lifecycle begins as a pipeline definition in the AWS Data Pipeline console or in a JSON file for the CLI. A pipeline definition must be validated and then it can be activated. At that point, the pipeline runs and schedules tasks. You can edit a running pipeline, re-activate the pipeline, then re-run the changed components. When you are done with your pipeline, you can delete it.

The complete lifecycle of a pipeline is shown in the following illustration.

Lifecycle of a Pipeline Task

The following diagram illustrates how AWS Data Pipeline and a task runner interact to process a scheduled task. A task is a discreet 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.
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.

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. 144).

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. 174)
  An DynamoDB table that contains data for HiveActivity (p. 207) or EmrActivity (p. 201) to use.
- MySqlDataNode (p. 179)
  A MySQL table and database query that represents data for a pipeline activity to use.
- RedshiftDataNode (p. 183)
  An Amazon Redshift table that contains data for RedshiftCopyActivity (p. 227) to use.
- S3DataNode (p. 187)
  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. 275)**: A JDBC database.
- **RdsDatabase (p. 276)**: An Amazon RDS database.
- **RedshiftDatabase (p. 277)**: 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. 196)**: Copies data from one location to another.
- **EmrActivity (p. 201)**: Runs an Amazon EMR cluster.
- **HiveActivity (p. 207)**: Runs a Hive query on an Amazon EMR cluster.
- **HiveCopyActivity (p. 212)**: Runs a Hive query on an Amazon EMR cluster with support for advanced data filtering and support for S3DataNode (p. 187) and DynamoDBDataNode (p. 174).
- **PigActivity (p. 218)**: Runs a Pig script on an Amazon EMR cluster.
- **RedshiftCopyActivity (p. 227)**: Copies data to and from Amazon Redshift tables.
- **ShellCommandActivity (p. 233)**: Runs a custom UNIX/Linux shell command as an activity.
- **SqlActivity (p. 239)**: 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. 43).

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.

**System-Managed Preconditions**

- **DynamoDBDataExists** (p. 256)
  Checks whether data exists in a specific DynamoDB table.
- **DynamoDBTableExists** (p. 259)
  Checks whether a DynamoDB table exists.
- **S3KeyExists** (p. 265)
  Checks whether an Amazon S3 key exists.
- **S3PrefixNotEmpty** (p. 268)
  Checks whether an Amazon S3 prefix is empty.

**User-Managed Preconditions**

- **Exists** (p. 262)
  Checks whether a data node exists.
- **ShellCommandPrecondition** (p. 271)
  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. 244)
  An EC2 instance that performs the work defined by a pipeline activity.
- **EmrCluster** (p. 250)
  An Amazon EMR cluster that performs the work defined by a pipeline activity, such as EmrActivity (p. 201).

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. 50).

**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. 46).

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. 289)**
  An action that sends an Amazon SNS notification to a Topic ARN based on certain events.
- **Terminate (p. 291)**
  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`.

For information about how to use Amazon SNS notifications, see Part One: Import Data into DynamoDB (p. 76).

Roles and Permissions

In AWS Data Pipeline, IAM roles determine what your pipeline can access and actions it can perform. Additionally, when your pipeline creates a resource, such as an EC2 instance, IAM roles determine the EC2 instance's permitted resources and actions. When you create a pipeline, you specify one IAM role to govern your pipeline and another IAM role to govern your pipeline's resources (referred to as a "resource role"); the same role can be used for both. Carefully consider the minimum permissions necessary for your pipeline to perform work, and define the IAM roles accordingly. It is important to note that even a modest pipeline might need access to resources and actions to various areas of AWS; for example:

- Accessing files in Amazon S3
- Creating and managing Amazon EMR clusters
- Creating and managing EC2 instances
- Accessing data in Amazon RDS or DynamoDB
- Sending notifications using Amazon SNS

When you use the console, AWS Data Pipeline creates the necessary IAM roles and policies, including a trusted entities list for you. However, CLI and SDK users must perform manual steps. For more information, see Setting Up IAM Roles (p. 4).

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. 292) 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:

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a name for the pipeline.</td>
</tr>
<tr>
<td>Description</td>
<td>(Optional) Enter a description for the pipeline.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
</table>
| Run         | • Choose **once on activation** to run the pipeline one time only. If you choose this option, all the other Schedule fields disappear.  
              • Choose **on schedule** to further specify parameters. |
| Run every   | Enter a period for every pipeline run.      |
| Starting    | Enter a time and date for the pipeline start time. Alternatively, your start date and time are automatically selected at pipeline activation. |
| Ending      | Enter a time and date for the pipeline start time. If you select **never** for the end date, your pipeline continues to execute indefinitely. |

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

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Choose this to have AWS Data Pipeline determine the roles for you.</td>
</tr>
</tbody>
</table>
| 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. |

**Time Series Style vs. Cron Style**

AWS Data Pipeline offers two types of 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.

In the CLI, **Time Series Style Scheduling** and **Cron Style Scheduling** are referred to as `timeseries`
and `cron` respectively. The default value for pipelines created using the CLI or SDK is `timeseries`. The
default value for pipelines created using the console is `cron`.

**Resources Ignore Schedule Type**

AWS Data Pipeline creates activity and data node instances at the beginning or end of the schedule in-
terval 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. It is possible when
the pipeline is set to **Time Series Style Scheduling** that AWS Data Pipeline creates resource instances
and sets them to `WAITING_ON_DEPENDENCIES` status much earlier than the activity or data nodes
start, where the amount of time is the length of the schedule interval.

**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 scheduledStartStart is earlier than 1 day ago. To override this behavior, use the `--force`
parameter from the CLI.

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 re-
source 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.
Time Zones

AWS Data Pipeline supports the date and time expressed in "YYYY-MM-DDTHH:MM:SS" format in UTC/GMT by default. For example, the following line 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"
```

Expressions let you create `DateTime` objects that use different time zones, including when you must take daylight savings time into account. The following expression uses a time zone.

```
#{inTimeZone(myDateTime,'America/Los_Angeles')}
```

Using the preceding expression results in the following `DateTime` value.

```
"2011-05-24T10:10:00 America/Los_Angeles"
```

AWS Data Pipeline uses the Joda Time API. For more information, go to http://joda-time.sourceforge.net/timezones.html.

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. 292).

Creating Pipelines

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. 21).
- 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 the AWS Data Pipeline command line interface (CLI) with a pipeline definition file in JSON format. For more information, see Creating a Pipeline Using the AWS Data Pipeline CLI (p. 314).
- Use an AWS SDK with a language-specific API. For more information, see Working with the API (p. 55).

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

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 and select a template from the dropdown menu. For more information about templates see the section called “Choose a Template” (p. 22).
   - If you use choose to use a template, parameters specific to that template will display. Provide values for the parameters as appropriate.
5. Choose whether to run the pipeline once on activation or on a schedule.
   - If you choose to run more than once:
     a. Choose a period for the pipeline. (Run every)
     b. Choose a Starting time and Ending time. AWS Data Pipeline uses the current activation time if you choose on pipeline activation. If you choose never for the end date and time, the pipeline runs indefinitely.

   **Note**
   If you do choose a finite interval for running the pipeline, it must be long enough to accommodate the period you selected in Step 5.a (p. 22).
6. Select an option for IAM Roles. If you select Default, DataPipeline 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 (p. 23)
- Run AWS CLI Command (p. 23)
- DynamoDB Cross Regional Table Copy (p. 23)
- Export DynamoDB Table to S3 (p. 24)
- Import DynamoDB Backup Data from S3 (p. 24)
- Run Job on an Elastic MapReduce Cluster (p. 25)
- Full Copy of RDS MySQL Table to S3 (p. 25)
• Incremental Copy of RDS MySQL Table to S3 (p. 25)
• Load S3 Data into RDS MySQL Table (p. 25)
• Full copy of RDS MySQL table to Redshift  (p. 26)
• Incremental copy of RDS MySQL table to Redshift  (p. 26)
• Load AWS Detailed Billing Report Into Redshift  (p. 27)
• Load Data from S3 Into Redshift  (p. 27)

**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.

**DynamoDB Cross Regional Table Copy**

The *DynamoDB Cross Regional Table Copy* AWS Data Pipeline template configures periodic movement of data between DynamoDB tables across regions or to a different table within the same region. This feature is useful in the following scenarios:

• Disaster recovery in the case of data loss or region failure
• Moving DynamoDB data across regions to support applications in those regions
• Performing full or incremental DynamoDB data backups

The template uses the following pipeline objects:

• DynamoDBDataNode
• HiveCopyActivity
• EmrCluster

The template uses the following pipeline objects:

• HiveCopyActivity (p. 212)
• EmrCluster (p. 250)
• DynamoDBDataNode (p. 174)
• DynamoDBExportDataFormat (p. 284)
The following diagram shows how this template copies data from an DynamoDB table in one region to an empty DynamoDB table in a different region. In the diagram, note that the destination table must already exist, with a primary key that matches the source table.

### Before the copy

**Region 1**

**Source DynamoDB Table**

<table>
<thead>
<tr>
<th>Id</th>
<th>Product</th>
<th>isDeleted</th>
<th>lastUpdated</th>
</tr>
</thead>
<tbody>
<tr>
<td>201</td>
<td>Bicycle</td>
<td>False</td>
<td>9/6/2013 12:33:09.22</td>
</tr>
<tr>
<td>202</td>
<td>Book</td>
<td>False</td>
<td>8/1/2013 5:18:44.91</td>
</tr>
<tr>
<td>203</td>
<td>Chair</td>
<td>True</td>
<td>8/14/2013 16:31:22.89</td>
</tr>
<tr>
<td>204</td>
<td>Lamp</td>
<td>False</td>
<td>11/12/2012 8:11:59.33</td>
</tr>
<tr>
<td>205</td>
<td>Shoes</td>
<td>False</td>
<td>9/5/2013 14:20:11.46</td>
</tr>
</tbody>
</table>

**Region 2**

**Destination DynamoDB Table**

<table>
<thead>
<tr>
<th>Id</th>
</tr>
</thead>
</table>

### After the copy

**Region 1**

**Source DynamoDB Table**

<table>
<thead>
<tr>
<th>Id</th>
<th>Product</th>
<th>isDeleted</th>
<th>lastUpdated</th>
</tr>
</thead>
<tbody>
<tr>
<td>201</td>
<td>Bicycle</td>
<td>False</td>
<td>9/6/2013 12:33:09.22</td>
</tr>
<tr>
<td>202</td>
<td>Book</td>
<td>False</td>
<td>8/1/2013 5:18:44.91</td>
</tr>
<tr>
<td>203</td>
<td>Chair</td>
<td>True</td>
<td>8/14/2013 16:31:22.89</td>
</tr>
<tr>
<td>204</td>
<td>Lamp</td>
<td>False</td>
<td>11/12/2012 8:11:59.33</td>
</tr>
<tr>
<td>205</td>
<td>Shoes</td>
<td>False</td>
<td>9/5/2013 14:20:11.46</td>
</tr>
</tbody>
</table>

**Region 2**

**Destination DynamoDB Table**

<table>
<thead>
<tr>
<th>Id</th>
<th>Product</th>
<th>isDeleted</th>
<th>lastUpdated</th>
</tr>
</thead>
<tbody>
<tr>
<td>201</td>
<td>Bicycle</td>
<td>False</td>
<td>9/6/2013 12:33:09.22</td>
</tr>
<tr>
<td>202</td>
<td>Book</td>
<td>False</td>
<td>8/1/2013 5:18:44.91</td>
</tr>
<tr>
<td>203</td>
<td>Chair</td>
<td>True</td>
<td>8/14/2013 16:31:22.89</td>
</tr>
<tr>
<td>204</td>
<td>Lamp</td>
<td>False</td>
<td>11/12/2012 8:11:59.33</td>
</tr>
<tr>
<td>205</td>
<td>Shoes</td>
<td>False</td>
<td>9/5/2013 14:20:11.46</td>
</tr>
</tbody>
</table>

### 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.

The template uses the following pipeline objects:

- **EmrActivity** (p. 201)
- **EmrCluster** (p. 250)
- **DynamoDBDataNode** (p. 174)
- **S3DataNode** (p. 187)

### 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.

The template uses the following pipeline objects:
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. 201)
- EmrCluster (p. 250)

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. 196)
- Ec2Resource (p. 244)
- MySqlDataNode (p. 179)
- S3DataNode (p. 187)

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. 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. 196)
- Ec2Resource (p. 244)
- MySqlDataNode (p. 179)
- S3DataNode (p. 187)

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. 196)
- Ec2Resource (p. 244)
- MySqlDataNode (p. 179)
- S3DataNode (p. 187)

**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
- MySqlDataNode
- 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. 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. 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. Please provide the column name that stores the last modified time value.

The template uses the following pipeline objects:

- RDSToS3CopyActivity
- CopyActivity
- RedshiftCopyActivity
- S3DataNode
- MySqlDataNode
- RedshiftDataNode
- RedshiftDatabase
Load AWS Detailed Billing Report Into Redshift

The Load AWS Detailed Billing Report Into Redshift template loads the AWS detailed billing report for the current month stored in an Amazon S3 folder to a Redshift table. If you would like to process files from previous months please pick a schedule that starts in the past. The input file must be of the .csv.zip format. Existing entries in the Redshift table are updated with data from Amazon S3 and new entries from Amazon S3 data are added to the Redshift table. If the table does not exist, it will be automatically created with the same schema as the AWS detailed billing report. The input report file is unzipped and converted to a GZIP file which is stored in the Amazon S3 staging folder before loading to Redshift.

The template uses the following pipeline objects:
- RDSToS3CopyActivity
- CopyActivity
- RedshiftCopyActivity
- S3DataNode
- MySqlDataNode
- 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
- MySqlDataNode
- RedshiftDataNode
- RedshiftDatabase
- Ec2Resource

Creating a Pipeline Using Parameterized Templates

You can parameterize templates by using myVariables or parameters within pipeline definitions. This allows you to keep a common pipeline definition but supply different parameters when you put the pipeline definition to create a new pipeline. This means you could use a tool to assemble a pipeline definition with different values for myVariables. Defining parameters also allows you to specify attributes which Data Pipeline will use to validate the values supplied for each myVariable. For example, the following pipeline definition:

```json
{
    "objects": [
        {
            "id": "ShellCommandActivityObj",
            "input": {
                "ref": "S3InputLocation"
            }
        }
    ]
}
```
specifies objects with variables (myShellCmd, myS3InputLoc, and myS3OutputLoc), which could be stored in a file, for example, file://pipelinedef.json. The following are parameter objects with attributes used for validation, which can be part of the pipeline definition or stored in a separate file (e.g. file://parameters.json):
The values can also be stored in the pipeline definition or in a file (e.g. `file://values.json`):

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

When you submit the pipeline definition using `PutPipelineDefinition`, you can supply the Objects, Parameters, and Values together. For example, in AWS CLI this would look like the following:

```
$ aws datapipeline create-pipeline --name myName --unique-id myUniqueId
{
   "pipelineId": "df-00123456ABC7DEF8HIJK"
}
$ aws datapipeline put-pipeline-definition --pipeline-id df-00123456ABC7DEF8HIJK
   --pipeline-definition file://pipelinedef.json --parameter-objects
   file://parameters.json --parameter-values file://values.json
$ aws datapipeline activate-pipeline --pipeline-id df-00123456ABC7DEF8HIJK
```

The following attributes are allowed for parameters:
Parameter Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Allowed value(s)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>Unique identifier of the parameter. Prefixing the id with an asterisk (*)—for example, *myVariable—will mask the value while it is typed or displayed. Furthermore, the value of *myVariable will be encrypted before it is stored by Data Pipeline.</td>
</tr>
<tr>
<td>type</td>
<td>String(default), Integer, Double, AWS::S3::ObjectKey</td>
<td>The parameter type that defines the allowed range of input values and validation rules.</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>String used to describe the parameter.</td>
</tr>
<tr>
<td>optional</td>
<td>Boolean (default is false)</td>
<td>Indicates whether parameter is optional or required.</td>
</tr>
<tr>
<td>allowedValues</td>
<td>List of Strings</td>
<td>Enumerates all permitted values for the parameter.</td>
</tr>
<tr>
<td>isArray</td>
<td>Boolean</td>
<td>Indicates whether the parameter is an array.</td>
</tr>
</tbody>
</table>

In addition to myVariables you can use Expressions. For more information, see Expressions (p. 162).

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. 32)
- 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.
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. In Pipeline Configuration, if you choose to enable logging, select a bucket in Amazon S3 to store logs for this pipeline.
6. Leave the Schedule fields set to their default values.
7. 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.
8. 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.
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.
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.
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.
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 validate and save your pipeline
1. On the pipeline page, click 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, click Close and then, in the right pane, click Errors/Warnings.
3. The Errors/Warnings pane lists the objects that failed validation. Click 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, click 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. On the List Pipelines page, in the Details column of your pipeline, click View pipeline.
2. In the pipeline page, click Activate.
3. In the confirmation dialog box, click 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.

You can view individual pipelines by clicking the arrow. This displays the schedule information and health status of pipeline activities. For more information about health status, see the section called “Interpreting Pipeline and Component Health State” (p. 36).

The following schedule-related activity fields are displayed in the console:

**Last Completed Run**
- The most recent completed component execution, not a scheduled execution. If today’s execution finished successfully, but yesterday’s execution took longer and overlapped with today’s execution, then yesterday’s run is the Last Completed Run because the completion time is more recent.

**Active Run**
- The current run of this activity. If the activity is not currently running, this has the value of Last Completed Run.

**Next Run**
- The next scheduled run for this activity.

To view your pipelines using the AWS CLI

Use the `list-pipelines` command as follows to list your pipelines.

```
aws datapipeline list-pipelines
```

To view your pipelines using the AWS Data Pipeline CLI
Use the `--list-pipelines` (p. 307) command as follows to list your pipelines:

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

## Interpreting Schedule Status Codes

The status levels displayed in the AWS Data Pipeline console and CLI indicate the condition of a pipeline and its components. Pipelines have a **SCHEDULED** status if they have passed validation and are ready, currently performing work, or done with their work. **PENDING** status means the pipeline is not able to perform work for some reason; for example, the pipeline definition might be incomplete or might have failed the validation step that all pipelines go through before activation. The pipeline status is simply an overview of a pipeline; to see more information, view the status of individual pipeline components.

Pipeline components have the following status values:

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

- **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` field defined by that component.

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

- **VALIDATING**
  - The pipeline definition is being validated by AWS Data Pipeline.

- **RUNNING**
  - The resource is running and ready to receive work.

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

- **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.

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

- **FINISHED**
  - The component completed its assigned work.

- **SHUTTING_DOWN**
  - The resource is shutting down after successfully completed its work.

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

- **CASCADE_FAILED**
  - The component or resource was canceled as a result of a cascade failure from one of its dependencies, but was probably not the original source of the failure.
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. 53).

**To view a pipeline definition using the console**

1. On the List Pipelines page, click on 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
```

**To get a pipeline definition using the AWS Data Pipeline CLI**

To get the complete pipeline definition, use the `--get` command. You can specify an output file to receive the pipeline definition. The default is to print the information to standard output (stdout).

The pipeline objects appear in alphabetical order, not in the order they appear in the pipeline definition file. The fields for each object are also returned in alphabetical order.

The following example prints the pipeline definition for the specified pipeline to a file named `output.json`.

```
datapipline --get --file output.json --id df-00627471SOVYZEXAMPLE
```

The following example prints the pipeline definition for the specified pipeline to stdout.

```
datapipline --get --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.

```
datapipeline --get --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. 155). For more information about troubleshooting failed or incomplete instance runs of your pipeline, see Resolving Common Problems (p. 156).

**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.

   ![Execution details](image)

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

   ![Inline summary](image)

   **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: 1A3456789ABC123, 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"
```

To monitor the progress of your pipeline using the AWS Data Pipeline 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.

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

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

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

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

```
datapipeline --list-runs --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:

---

**Editing Your Pipelines**

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.

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:

- You can't change the default objects
- You can't change the schedule of an object
- You can't change the dependencies between objects
- You can't add, delete, or modify reference fields for existing objects; only non-reference fields are allowed
- You can't edit pipelines in the FINISHED state.
- New objects cannot reference an previously existing object for the output field; previously existing objects are only allowed in the input field
- Changes only apply to new instances of pipeline objects

**To edit an active pipeline using the console**

1. On the List Pipelines page, check the Pipeline ID and Name columns for your pipeline and 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 Pipelines (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.

---

**To edit an active pipeline using the AWS CLI**

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

To edit an active pipeline using the AWS Data Pipeline CLI

First, download a copy of the current pipeline definition using the **--get, --g** (p. 305) 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 a file named **output.txt**.

```
datapipeline --get --file output.txt --id df-00627471SOVYZEXAMPLE
```

Edit the pipeline definition as needed and save it as **my-updated-file.txt**. Update your pipeline definition using the **--put** (p. 310) command. The following example uploads the updated pipeline definition file.

```
datapipeline --put my-updated-file.txt --id df-00627471SOVYZEXAMPLE
```

You can retrieve the pipeline definition again using the **--get** command to ensure that the update was successful. To active the pipeline, use the **--activate** (p. 300) command. To re-run one or more pipeline components, use the **--rerun** (p. 311) command.
Cloning Your Pipelines

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 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 Activate.

Deleting Your Pipelines

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 Delete.
3. In the confirmation dialog box, click Delete to confirm the delete request.

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
```

To delete a pipeline using the AWS Data Pipeline CLI

To delete a pipeline, use the --delete (p. 304) command. The following command deletes the specified pipeline.

```
datapipeline --delete --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. 233).

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 `INPUTx_STAGING_DIR` and output data is available as `OUTPUTx_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.
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.
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. 165).

```
{  
  "id": "MyHiveActivity",  
  "type": "HiveActivity",  
  "schedule": {  
    "ref": "MySchedule"  
  },  
  "runsOn": {  
    "ref": "MyEmrResource"  
  },  
  "input": {  
    "ref": "MyInputData"  
  },  
  "output": {  
    "ref": "MyOutputData"  
  },  
  "hiveScript": "INSERT OVERWRITE TABLE ${output1} select * from ${input1};"  
}
```
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 (p. 47)
- Set Up Connectivity Between Resources (p. 47)
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.
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 Amazon-ProvidedDNS 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. 250), 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.
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. 244), 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.
   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.
   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.
   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.

```json
{
    "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. 250).

**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.

```json
{
    "id" : "MyEC2Resource",
    "type" : "Ec2Resource",
    "actionOnTaskFailure" : "terminate",
    "actionOnResourceFailure" : "retryAll",
    "maximumRetries" : "1",
    "role" : "test-role",
    "resourceRole" : "test-role",
    "instanceType" : "m1.medium",
    "securityGroupIds" : "sg-12345678",
    "subnetId": "subnet-la2b3c4d",
    "associatePublicIpAddress": "true",
    "keyPair" : "my-key-pair"
}
```

For more information, see Ec2Resource (p. 244).
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.
2. In the List Pipelines page, select the pipeline.
3. In the Resources pane, in the EmrCluster section, set the taskInstanceBidPrice field to your Spot Instance bid price. The taskInstanceBidPrice value is the maximum dollar amount for your spot instance bid and is a decimal value between 0 and 20.00 exclusive.

   Note
   When you set the taskInstanceBidPrice value, you must also provide values for coreInstanceCount and taskInstanceCount. For more information, see EmrCluster (p. 250).

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 (eu-west-1), 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.

```json
{
    "objects": [
    {
        "id": "Hourly",
        "type": "Schedule",
        "startDateTime": "2012-11-19T07:48:00",
        "endDate": "2012-11-21T07:48:00",
        "period": "1 hours"
    },
    {
        "id": "MyCluster",
        "type": "EmrCluster",
        "masterInstanceType": "ml.small",
        "region": "eu-west-1",
        "schedule": {
```
Cascading Failures and Reruns

AWS Data Pipeline allows you to configure the way pipeline objects behave when a dependency fails or is cancelled 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 `CANCELLED`.
- Any objects that are already `FINISHED`, `FAILED`, or `CANCELLED` are ignored.

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

<table>
<thead>
<tr>
<th>Region Name</th>
<th>Region Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>US East (N. Virginia) region</td>
<td>us-east-1</td>
</tr>
<tr>
<td>US West (N. California) region</td>
<td>us-west-1</td>
</tr>
<tr>
<td>US West (Oregon) region</td>
<td>us-west-2</td>
</tr>
<tr>
<td>EU (Ireland) region</td>
<td>eu-west-1</td>
</tr>
<tr>
<td>Asia Pacific (Tokyo) region</td>
<td>ap-northeast-1</td>
</tr>
<tr>
<td>Asia Pacific (Singapore) region</td>
<td>ap-southeast-1</td>
</tr>
<tr>
<td>Asia Pacific (Sydney) region</td>
<td>ap-southeast-2</td>
</tr>
<tr>
<td>South America (Sao Paulo) region</td>
<td>sa-east-1</td>
</tr>
</tbody>
</table>
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 will not 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 `CANCELLED` 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 `CANCELLED` 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 `CANCELLED` state.

If you attempt to rerun a `CASCADE_FAILED` object and any of its dependencies are `FAILED`, `CASCADE_FAILED`, or `CANCELLED`, 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. 161).

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. 173).

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. 161). For more information about functions that you can use to evaluate field values, see Expression Evaluation (p. 165).
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. 173).

```json
{
   "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".

```json
{
   "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.

```json
{
   "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:

```json
{
   "id": "S3DataInput",
   "type": "S3DataNode",
   "schedule": {"ref": "TheSchedule"},
   "filePath": "s3://bucket_name",
   "myCustomField": "value",
   "my_customFieldReference": "value"
}
```
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 AWS Data Pipeline (p. 3).

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. 55).

#### 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.

```json
{
  "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 K2QH8DN0U907N97FNA2GDL8OBVV4KQNS05AEMVJF66Q9ASJAAG.
- **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"}
```
The following tutorials walk you step-by-step through the process of creating and using pipelines with AWS Data Pipeline.

**Tutorials**
- Process Access Logs Using Amazon EMR with Hive (p. 59)
- Process Data Using Amazon EMR to Run a Hadoop Streaming Cluster (p. 66)
- Import and Export DynamoDB Data (p. 75)
- Copy CSV Data from Amazon S3 to Amazon S3 (p. 91)
- Export MySQL Data to Amazon S3 with CopyActivity (p. 103)
- Copying DynamoDB Data Across Regions (p. 116)
- Copy Data to Amazon Redshift Using AWS Data Pipeline (p. 131)

**Process Access Logs Using Amazon EMR with Hive**

This tutorial shows you how to use the AWS Data Pipeline console to create a pipeline that uses an Amazon EMR cluster and a Hive script to read access logs, select certain columns, and write the reformatted output to an Amazon S3 bucket. This tutorial uses a console template and omits optional steps to get you started with AWS Data Pipeline as quickly as possible.

**Prerequisites**

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

**Tasks**
- Create the Pipeline (p. 60)
- Choose the Template (p. 60)
- Complete the Fields (p. 60)
- Save and Activate Your Pipeline (p. 65)
- View the Running Pipeline (p. 65)
- Verify the Output (p. 65)
Create the Pipeline

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

To create the pipeline definition

1. Open the AWS Data Pipeline console.
2. Click either Create new pipeline or Get started now (if you haven't created a pipeline in this region).
3. In Name, enter a name (for example, Log Processing with Hive).
4. In Description, enter a description.
5. Leave the Schedule fields set to their default values.
6. In Pipeline Configuration, leave logging enabled and use the Amazon S3 file widget to select a bucket to store all of your pipeline logs.
7. Leave IAM roles set to Default.
8. Click Create.

Choose the Template

Templates are pre-defined pipelines that you can modify to suit your needs.

To choose the template

1. On the pipeline screen, click Templates and select Run Hive analytics on S3 data.
2. The AWS Data Pipeline console pre-populates a pipeline with the base objects necessary, such as the S3DataNodes, EmrCluster, HiveActivity, etc., as shown in the following screen.

Complete the Fields

Templates are pre-populated with the commonly required fields and you complete the rest. Review the fields and values described in this section and provide any missing information.

Contents

- HiveActivity (p. 61)
- S3DataNodes (p. 61)
HiveActivity

The activity brings together the data nodes, schedule, and computational resources, and defines the work to perform.

In the right pane, click Activities and review the template-provided values for the activity. Note the Hive script set to run: INSERT OVERWRITE TABLE ${output1} select host,user,time,requested,status,size from ${input1};

S3DataNodes

The data nodes define the storage points for pipeline data, for example the source and destination data directories. In the right pane, click DataNodes and review the template values for the data nodes. Note the MyInputData and MyOutput data nodes point to Amazon S3 and use custom data formats.

To set the input path to the sample Apache web logs

1. Under the MyInputData section, click Add an optional field .. and select Directory Path.
To set the output to your Amazon S3 bucket

1. Under the **MyOutputData** section, click **Add an optional field** .. and select **Directory Path**.
2. For **Directory Path**, enter the path to your Amazon S3 bucket; for example `s3://my bucket name/folder`. For more information, see Create a Bucket in the Amazon Simple Storage Service Getting Started Guide.

**Schedule**

The schedule determines when activities start and how often they run.

**To set the schedule**

1. In the right pane, click **Schedules**.
2. Under **MyEmrResourcePeriod**, click **Start Date Time** and set the value to the current date.
3. Under **MyEmrResourcePeriod**, click **Period** and set the value to **1 Hour**.
4. Click **Start Date Time** and select today's date and select a time that is very early in the day, such as **00:30:00**. Time values in the AWS Data Pipeline console are in UTC format.
5. Click **End Date Time** and select the same day and select the time to be 1 hour later than the start time, such as **01:30:00**. With the period set to **1 Hour**, this ensures that the pipeline executes only one time.
EmrCluster

The resource defines what computational resource performs the work that the activity specifies, such as an Amazon EMR cluster.

To set a valid key pair and debug log

1. In the right pane, click Resources.
2. Under MyEmrResource, click the Key Pair field and replace the test-pair value by typing the name of a valid Amazon EC2 key pair in your account. For more information, see Amazon EC2 Key Pairs.
3. Click Add an optional field again and select Emr Log Uri.
4. For Emr Log Uri, type the path to your own Amazon S3 bucket; for example s3://my bucket name/folder. You must set this value to debug Amazon EMR cluster errors if they occur.

Custom Data Types

The custom data type defines the format of data that the pipeline activity reads and writes.

To review the input data format

1. In the right pane, click Others.
2. Review the fields and note that the RegEx type set for MyInputDataType component uses the Java formatter syntax. For more information, see Format String Syntax.
To set custom record and column separators

- Under MyOutputDataType, type \n (newline) for the Record Separator and type \t (tab) for the Column Separator as shown.
Save and Activate Your Pipeline

As soon as you save your pipeline definition, AWS Data Pipeline looks for syntax errors and missing values in your pipeline definition. You must fix the errors in the pipeline definition before activating your pipeline. If you encounter an error message, see Resolving Common Problems (p. 156).

To validate and save your pipeline

- On the Pipeline: name of your pipeline page, click Save Pipeline.

To activate your pipeline

1. Click Activate.

   A confirmation dialog box opens up confirming the activation.

   Important
   Your pipeline is running and is incurring charges. For more information, see AWS Data Pipeline pricing. If you would like to stop incurring the AWS Data Pipeline usage charges, delete your pipeline.

2. Click Close.

View the Running Pipeline

You can view the pipeline details to ensure that the pipeline starts correctly.

To monitor the progress of your pipeline

1. On the Pipeline: screen, click Back to List of Pipelines.
2. On the List Pipelines page, click on the triangle button next to your pipeline and click on Execution Details.
3. The Execution details page lists the status of each component in your pipeline definition. You can click Update or press F5 to update the pipeline status display.

   Note
   If you do not see runs listed, ensure that the Start (in UTC) and End (in UTC) filter fully encompasses your pipeline instance Scheduled Start and Scheduled End dates and times. Then click Update.

4. When the Status column of all the objects in your pipeline indicates FINISHED, your pipeline has successfully completed the copy activity.

5. If you see an ERROR health status or a FAILED status for any instances, or the instance status does not progress beyond WAITING_FOR_RUNNER and WAITING_ON_DEPENDENCIES, troubleshoot your pipeline settings. For more information about troubleshooting any failed or incomplete instance runs of your pipeline, see Resolving Common Problems (p. 156).

Verify the Output

After the Amazon EMR cluster starts and processes the logs, the output shows up in the output folder that you specified earlier.
To verify the output file

1. Navigate to your output Amazon S3 bucket and open the output file using your preferred text editor. The output file name is a GUID plus a numeric value but no file extension, such as f8cc485d-e926-4b48-9729-9b0ce0f15984_000000.

2. Review the fields and confirm that the source data appears in the format specified by the MyOutput-Datatype component as shown.

3. Ensure that your output appears as expected with no error messages. For more information about troubleshooting any failed or incomplete instance runs of your pipeline, see Resolving Common Problems (p. 156).

Process Data Using Amazon EMR to Run a Hadoop Streaming Cluster

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.

This tutorial uses the following pipeline objects:

EmrActivity
   The EmrActivity defines the work to perform in the pipeline.

   This tutorial uses the EmrActivity to run a pre-existing Hadoop Streaming job provided by Amazon EMR.

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

Resource
   Resource AWS Data Pipeline must use to perform this activity.

   This tutorial uses EmrCluster, a set of Amazon EC2 instances, provided by AWS Data Pipeline. AWS Data Pipeline automatically launches the Amazon EMR cluster and then terminates the cluster after the task finishes.

Action
   Action AWS Data Pipeline must take when the specified conditions are met.
This tutorial uses the SnsAlarm action to send Amazon SNS notification to the Amazon SNS topic you specify, after the task finishes successfully.

**Before You Begin**

Be sure you've completed the following steps.

- Complete the tasks in **Setting Up AWS Data Pipeline** (p. ?).
- (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. 46).
- Create an Amazon SNS 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.

**Note**

Some of the actions described in this tutorial can generate AWS usage charges, depending on whether you are using the AWS Free Usage Tier.

**Using the AWS Data Pipeline Console**

To create a pipeline using the AWS Data Pipeline console, complete the following tasks.

**Tasks**

- Create and Configure the Pipeline Objects (p. 67)
- Save and Validate Your Pipeline (p. 69)
- Verify Your Pipeline Definition (p. 70)
- Activate your Pipeline (p. 70)
- Monitor the Progress of Your Pipeline Runs (p. 70)
- (Optional) Delete your Pipeline (p. 71)

**Create and Configure the Pipeline Objects**

First, complete the initial pipeline creation screen.

**To create your pipeline**

1. Open the AWS Data Pipeline console.
2. Click either Create new pipeline or Get started now (if you haven't created a pipeline in this region).
3. On the Create Pipeline page:
   a. In Name, enter a name (for example, MyEMRJob).
   b. In Description, enter a description.
   c. Leave the Schedule fields set to their default values for this tutorial.
   d. Leave IAM roles set to its default value, which is to use the default IAM roles, DataPipelineDefaultRole for the pipeline role and DataPipelineDefaultResourceRole for the resource role.
   e. Click Create.
Next, add an activity to your pipeline and the objects that AWS Data Pipeline must use to perform this activity.

**To configure the activity**

1. On the pipeline page, click **Add activity**.
2. In the **Activities** pane
   a. Enter the name of the activity; for example, **MyEMRActivity**
   b. In **Type**, select **EmrActivity**.
   c. In **Schedule**, select **Create new: Schedule**.
   d. 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
   ```
   e. In **Add an optional field**, select **Runs On**.
   f. In **Runs On**, select **Create new: EmrCluster**.
   g. In **Add an optional field**, select **On Success**.
   h. In **On Success**, select **Create new: Action**.

The pipeline pane shows a single activity icon for this pipeline.

Next, configure run date and time for your pipeline.

**To configure the schedule**

1. On the pipeline page, in the right pane, click **Schedules**.
2. In the **Schedules** pane:
   a. Enter a schedule name for this activity (for example, **MyEMRJobSchedule**).
   b. In **Start Date Time**, select the date from the calendar, and then enter the time to start the activity.

   **Note**

   AWS Data Pipeline supports the date and time expressed in UTC format only.

   c. In **Period**, enter the amount of time between pipeline runs (for example, 1), and then select the period category (for example, **Days**).
   d. (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 start your pipeline immediately, set **Start Date Time** to a past date.

   **Important**

   If you set a start date far in the past, it can cause multiple instances of the pipeline to launch simultaneously as AWS Data Pipeline attempts to catch up to the backlog of work. For more information, see **Backfill Tasks** (p. 20).

Next, configure the resource AWS Data Pipeline must use to perform the Amazon EMR job.
To configure the resource

1. On the pipeline page, in the right pane, click Resources.
2. In the Resources pane:
   a. In Name, enter the name for your Amazon EMR cluster (for example, MyEMRCluster).
   b. Leave Type set to EmrCluster.
   c. In Add an optional field, select Subnet Id.
   d. In Subnet Id, enter the ID of the subnet.
   e. In Schedule, select MyEMRJobSchedule.
   f. In 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, for example prototyping and troubleshooting.

   g. In Add an optional field, select Emr Log Uri. Set the value to an Amazon S3 bucket to store your Amazon EMR logs for troubleshooting. For example, s3://examples-bucket/emrlogs.

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

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

To configure the Amazon SNS notification action

1. On the pipeline page, in the right pane, click Others.
2. In the Others pane:
   a. In DefaultAction1 Name, enter the name for your Amazon SNS notification (for example, MyEMRJobNotice).
   b. In Type, select SnsAlarm.
   c. In Subject, enter the subject line for your notification.
   d. Leave the entry in Role set to default.
   e. In Topic Arn, enter the ARN of your Amazon SNS topic.
   f. In Message, enter the message content.

You have now completed all the steps required for creating your pipeline definition. Next, save your pipeline.

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 validate and save your pipeline

1. On the pipeline page, click **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, click **Close** and then, in the right pane, click **Errors/Warnings**.
3. The **Errors/Warnings** pane lists the objects that failed validation. Click 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, click **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, check if your newly-created pipeline is listed.
   - AWS Data Pipeline has created a unique **Pipeline ID** for your pipeline definition.
   - The **Status** column in the row listing your pipeline should show **PENDING**.
2. Click on the triangle icon next to your pipeline. The **Pipeline summary** pane below shows the details of your pipeline runs. Because your pipeline is not yet activated, you should see only 0s at this point.
3. In the **Pipeline summary** pane, click **View fields** to see the configuration of your pipeline definition.
4. Click **Close**.

**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. On the **List Pipelines** page, in the **Details** column of your pipeline, click **View pipeline**.
2. In the pipeline page, click **Activate**.
3. In the confirmation dialog box, click **Close**.

**Monitor the Progress of Your Pipeline Runs**

You can monitor the progress of your pipeline. For more information about instance status, see Interpreting Pipeline Status Details (p. 155). For more information about troubleshooting failed or incomplete instance runs of your pipeline, see Resolving Common Problems (p. 156).
To monitor the progress of your pipeline using the console

1. On the **List Pipelines** page, in the **Details** column for your pipeline, click **View instance details**.
2. The **Instance details** page lists the status of each instance.

   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 instances 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 instances 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.

   a. Click the triangle next to an instance.
   b. In the **Instance summary** pane, click **View instance fields** to see the fields associated with the selected instance. If the status of the instance is **FAILED**, the details box has an entry indicating the reason for failure. For example, @failureReason = Resource not healthy terminated.
   c. In the **Instance summary** pane, in the **Select attempt for this instance** box, select the attempt number.
   d. In the **Instance summary** pane, click **View attempt fields** to see details of fields associated with the selected attempt.

5. To take an action on your incomplete or failed instance, select an action (**Rerun|Cancel|Mark Finished**) from the **Action** column of the instance.

**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 using the console**

1. In the **List Pipelines** page, click the check box next to your pipeline.
2. Click **Delete**.
3. In the confirmation dialog box, click **Delete** to confirm the delete request.
Using the Command Line Interface

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. Select, install, and configure a CLI. For more information, see (Optional) Installing a Command Line Interface (p. 3).
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, you must create these roles manually. For more information, see Setting Up IAM Roles (p. 4).

Tasks

- Creating the Pipeline Definition File (p. 72)
- Uploading and Activating the Pipeline Definition (p. 73)
- Monitoring the Pipeline (p. 74)

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.

```json
{
  "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"
      }
    }
  ],
}
```
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. 153).

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

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

**Monitoring the Pipeline**

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

These tutorials demonstrates 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. 76)
- Part Two: Export Data from DynamoDB (p. 84)

These tutorials involve the following concepts and procedures:
- Using the AWS Data Pipeline console and command-line interface (CLI) to create and configure pipelines
- Creating and configuring DynamoDB tables
- Creating and allocating work to Amazon EMR clusters
- Querying and processing data with Hive scripts
- Storing and accessing data using Amazon S3
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. 76)
- Start Import from the DynamoDB Console (p. 77)
- Create the Pipeline using the AWS Data Pipeline Console (p. 78)
- Choose the Template (p. 78)
- Complete the Fields (p. 79)
- Validate and Save Your Pipeline (p. 81)
- Activate your Pipeline (p. 82)
- Monitor the Progress of Your Pipeline Runs (p. 82)
- Verify Data Import (p. 83)
- (Optional) Delete your Pipeline (p. 84)

Before You Begin

Be sure you've completed the following steps.

- Complete the tasks in Setting Up AWS Data Pipeline (p. ?).
- (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. 46).
- 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.
- Create an Amazon SNS 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. If you already have an Amazon SNS topic ARN to which you have subscribed, you can skip this step.
- Create an DynamoDB table to store data as defined by the following procedure.

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 an DynamoDB Table

This section explains how to create an DynamoDB table that is a prerequisite for this tutorial. For more information, see Working with Tables in DynamoDB in the DynamoDB Developer Guide.

Note
If you already have a DynamoDB table, you can skip this procedure to create one.

To create a DynamoDB table

1. Open the DynamoDB console.
2. Click Create Table.
3. On the Create Table / Primary Key page, enter a name (for example, MyTable) in Table Name.
   
   Note
   Your table name must be unique.
4. In the Primary Key section, for the Primary Key Type radio button, select Hash.
5. In the Hash Attribute Name field, select Number and enter the string Id.
6. Click Continue.
7. On the Create Table / Provisioned Throughput Capacity page, in Read Capacity Units, enter 5.
8. In Write Capacity Units, enter 5.
   
   Note
   In this example, we use read and write capacity unit values of five because the sample input data is small. You may need a larger value depending on the size of your actual input data set. For more information, see Provisioned Throughput in Amazon DynamoDB in the Amazon DynamoDB Developer Guide.
9. Click Continue.
10. On the Create Table / Throughput Alarms page, in Send notification to, enter your email address.

Start Import from the DynamoDB Console

You can begin the DynamoDB import operation from within the DynamoDB console.

To start the data import

1. Open the DynamoDB console.
2. On the Tables screen, click your DynamoDB table and click the Import Table button.
3. On the Import Table screen, read the walkthrough and check I have read the walkthrough, then select Build a Pipeline. This opens the AWS Data Pipeline console so that you can choose a template to import the DynamoDB table data.
Create the Pipeline using the AWS Data Pipeline Console

To create the pipeline

1. Open the AWS Data Pipeline console, or arrive at the AWS Data Pipeline console through the Build a Pipeline button in the DynamoDB console.
2. Click either Create new pipeline or Get started now (if you haven’t created a pipeline in this region).
3. On the Create Pipeline page:
   a. In Name, enter a name (for example, CopyMyS3Data).
   b. In Description, enter a description.
   c. Choose whether to run the pipeline once on activation or on a schedule.
   d. Leave IAM roles set to its default value, which is to use the default IAM roles, DataPipelineDefaultRole for the pipeline role and DataPipelineDefaultResourceRole for the resource role.

   Note
   If you have created your own custom IAM roles and would like to use them in this tutorial, you can select them now.
   e. Click Create.

Choose the Template

On the Pipeline screen, click Templates and select Export S3 to DynamoDB. The AWS Data Pipeline console pre-populates a pipeline definition template with the base objects necessary to import data from Amazon S3, as shown in the following screen.
Review the template and complete the missing fields. You start by choosing the schedule and frequency by which you want your data export operation to run.

**To configure the schedule**

- On the pipeline page, click **Schedules**.
  
  a. In the **ImportSchedule** section, set **Period** to **1 Hours**.
  
  b. Set **Start Date Time** using the calendar to the current date, such as **2012-12-18** and the time to **00:00:00 UTC**.
  
  c. In **Add an optional field**, select **End Date Time**.
  
  d. Set **End Date Time** using the calendar to the following day, such as **2012-12-18** and the time to **01:00:00 UTC**.

**Important**

Avoid creating a recurring pipeline that imports the same data multiple times. The above steps schedule an import job to run once starting at 2012-12-18 01:00:00 UTC. If you prefer a recurring import, extend Start Date Time or End Date Time to include multiple periods and use a date based expression like `s3://myBucket/#{@scheduledStartTime}` in `MyS3Data.DirectoryPath` to specify a separate directory path for each period. When a schedule's startDateTime is in the past, AWS Data Pipeline will backfill your pipeline and begin scheduling runs immediately beginning at startDateTime. For more information, see **Schedule (p. 292)**.

**Complete the Fields**

Templates are pre-populated with the commonly required fields and you complete the rest. Review the fields and values described in this section and provide any missing information.

**Contents**

- DataNodes (p. 79)
- EmrCluster (p. 80)
- EmrActivity (p. 81)
- Notifications (p. 81)

**DataNodes**

Next, complete the data node objects in your pipeline definition template.

**To configure the DynamoDB data node**

1. On the pipeline page, select **DataNodes**.
2. In the **DataNodes** pane:
   
   a. In the **MyDynamoDBData** section, in **Table Name**, type the name of the DynamoDB table where you want to store the output data; for example: ```MyTable```.

For a complete list of fields, see **DynamoDBDataNode (p. 174)**.
To configure the Amazon S3 data node

- In the **DataNodes** pane:
  - In the **MyS3Data** section, in **Directory Path**, type a valid Amazon S3 directory path for the location of your source data, for example, `s3://elasticmapreduce/samples/Store/Product-Catalog`. This sample, fictional product catalog that is pre-populated with delimited data for demonstration purposes. **DirectoryPath** points to either a directory containing multiple files to import, or a path to one specific file to import.

For a complete list of fields, see [S3DataNode](#) (p. 187).

**EmrCluster**

Next, you complete the resources that will run the data import activities. Many of the fields are auto-populated by the template, as shown in the following screen. You only need to complete the empty fields.

To configure the resources

1. On the pipeline page, in the right pane, select **Resources**.

   **Note**
   For more information about how to identify which Amazon EMR cluster serves your pipeline, see [Identifying the Amazon EMR Cluster that Serves Your Pipeline](#) (p. 154).

2. In the **Resources** pane:
   a. In **Emr Log Uri**, type the path where to store Amazon EMR debugging logs, using the Amazon S3 bucket that you configured in part one of this tutorial; for example: `s3://my-test-buck- et/emr_debug_logs`.
   b. [EC2-VPC] In **Add an optional field**, select **Subnet Id**.
c. [EC2-VPC] In Subnet Id, enter the ID of the subnet.

For a complete list of fields, see EmrCluster (p. 250).

**EmrActivity**

Next, complete the activity that represents the steps to perform in your data import operation.

**To configure the activity**

1. On the pipeline page, select Activities.
2. In the MyImportJob section, review the default options already provided. You are not required to manually configure any options in this section.

   **Note**
   Consider updating myDynamoDBWriteThroughputRatio. It sets the rate of write operations to keep your DynamoDB provisioned throughput rate in the allocated range for your table. The value is between 0.1 and 1.5, inclusively. For more information, see Specifying Read and Write Requirements for Tables.

For a complete list of fields, see EmrActivity (p. 201).

**Notifications**

Next, configure the SNS notification action AWS Data Pipeline must perform depending on the outcome of the activity.

**To configure the SNS success, failure, and late notification action**

1. On the pipeline page, in the right pane, click Others.
2. In the Others pane:
   a. In the LateSnsAlarm section, in Topic Arn, enter the ARN of your Amazon SNS topic that you created earlier in the tutorial; for example: arn:aws:sns:us-east-1:403EXAMPLE:my-example-topic
   b. In the FailureSnsAlarm section, in Topic Arn, enter the ARN of your Amazon SNS topic that you created earlier in the tutorial; for example: arn:aws:sns:us-east-1:403EXAMPLE:my-example-topic
   c. In the SuccessSnsAlarm section, in Topic Arn, enter the ARN of your Amazon SNS topic that you created earlier in the tutorial; for example: arn:aws:sns:us-east-1:403EXAMPLE:my-example-topic

For a complete list of fields, see SnsAlarm (p. 289).

**Validate and Save 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 validate and save your pipeline

1. On the pipeline page, click **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, click **Close** and then, in the right pane, click **Errors/Warnings**.
3. The Errors/Warnings pane lists the objects that failed validation. Click 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, click **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. On the List Pipelines page, in the Details column of your pipeline, click **View pipeline**.
2. In the pipeline page, click **Activate**.
3. In the confirmation dialog box, click **Close**.

**Monitor the Progress of Your Pipeline Runs**

You can monitor the progress of your pipeline. For more information about instance status, see Interpreting Pipeline Status Details (p. 155). For more information about troubleshooting failed or incomplete instance runs of your pipeline, see Resolving Common Problems (p. 156).

**To monitor the progress of your pipeline using the console**

1. On the List Pipelines page, in the Details column for your pipeline, click **View instance details**.
2. The Instance details page lists the status of each instance.
   
   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 instances 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 instances 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.
   
   a. Click the triangle next to an instance.
   b. In the Instance summary pane, click **View instance fields** to see the fields associated with the selected instance. If the status of the instance is **FAILED**, the details box has an entry indic-
ating the reason for failure. For example, @failureReason = Resource not healthy terminated.

c. In the Instance summary pane, in the Select attempt for this instance box, select the attempt number.

d. In the Instance summary pane, click View attempt fields to see details of fields associated with the selected attempt.

5. To take an action on your incomplete or failed instance, select an action (Rerun, Cancel, Mark Finished) from the Action column of the instance.

Verify 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 the Explore Table button.
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.

<table>
<thead>
<tr>
<th>Id</th>
<th>Price</th>
<th>ProductCategory</th>
<th>Title</th>
<th>Brand</th>
</tr>
</thead>
<tbody>
<tr>
<td>205</td>
<td>500</td>
<td>&quot;Bicycle&quot;</td>
<td>&quot;20-Bike-205&quot;</td>
<td>&quot;Brand-Comp&quot;</td>
</tr>
<tr>
<td>203</td>
<td>300</td>
<td>&quot;Bicycle&quot;</td>
<td>&quot;19-Bike-203&quot;</td>
<td>&quot;Brand-Comp&quot;</td>
</tr>
<tr>
<td>202</td>
<td>200</td>
<td>&quot;Bicycle&quot;</td>
<td>&quot;21-Bike-202&quot;</td>
<td>&quot;Brand-Comp&quot;</td>
</tr>
<tr>
<td>201</td>
<td>100</td>
<td>&quot;Bicycle&quot;</td>
<td>&quot;18-Bike-201&quot;</td>
<td>&quot;Mountain A&quot;</td>
</tr>
<tr>
<td>204</td>
<td>400</td>
<td>&quot;Bicycle&quot;</td>
<td>&quot;19-Bike-204&quot;</td>
<td>&quot;Brand-Comp&quot;</td>
</tr>
<tr>
<td>102</td>
<td>20</td>
<td>&quot;Book&quot;</td>
<td>&quot;Book 102 Title&quot;</td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>2000</td>
<td>&quot;Book&quot;</td>
<td>&quot;Book 103 Title&quot;</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>2</td>
<td>&quot;Book&quot;</td>
<td>&quot;Book 101 Title&quot;</td>
<td></td>
</tr>
</tbody>
</table>
(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 using the console

1. In the List Pipelines page, click the check box next to your pipeline.
2. Click Delete.
3. In the confirmation dialog box, click Delete to confirm the delete request.

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.

This tutorial involves the following concepts and procedures:

- Using the AWS Data Pipeline console and command-line interface (CLI) to create and configure pipelines
- Creating and configuring DynamoDB tables
- Creating and allocating work to Amazon EMR clusters
- Querying and processing data with Hive scripts
- Storing and accessing data using Amazon S3

Tasks

- Before You Begin (p. 84)
- Start Export from the DynamoDB Console (p. 85)
- Create the Pipeline using the AWS Data Pipeline Console (p. 85)
- Choose the Template (p. 86)
- Complete the Fields (p. 87)
- Validate and Save Your Pipeline (p. 89)
- Activate your Pipeline (p. 89)
- Monitor the Progress of Your Pipeline Runs (p. 90)
- Verify Data Export File (p. 91)
- (Optional) Delete your Pipeline (p. 91)

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. 76).

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

- Complete the tasks in Setting Up AWS Data Pipeline (p. ?).
- Create an Amazon S3 bucket as a data output location. For more information, see Create a Bucket in the Amazon Simple Storage Service Getting Started Guide.
- Create an Amazon SNS 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. If you already have an Amazon SNS topic ARN to which you have subscribed, you can skip this step.

- 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. 76).

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.

Start Export from the DynamoDB Console

You can begin the DynamoDB export operation from within the DynamoDB console.

To start the data export

1. Open the DynamoDB console.
2. On the Tables screen, click your DynamoDB table and click the Export Table button.
3. On the Import / Export Table screen, select Build a Pipeline. This opens the AWS Data Pipeline console so that you can choose a template to export the DynamoDB table data.

Create the Pipeline using the AWS Data Pipeline Console

To create the pipeline

1. Open the AWS Data Pipeline console, or arrive at the AWS Data Pipeline console through the Build a Pipeline button in the DynamoDB console.
2. Click Create new pipeline.
3. On the Create Pipeline page:
   a. In Name, enter a name (for example, CopyMyS3Data).
b. In **Description**, enter a description.
c. Choose whether to run the pipeline once on activation or on a schedule.
d. Leave **IAM role** set to its default value for this tutorial, which uses **DataPipelineDefaultRole** for the pipeline role and **DataPipelineDefaultResourceRole** for the resource role.

**Note**

If you have created your own custom IAM roles and would like to use them in this tutorial, you can select them now.
e. Click **Create**.

Choose the Template

On the Pipeline screen, click **Templates** and select **Export DynamoDB to S3**. The AWS Data Pipeline console pre-populates a pipeline definition template with the base objects necessary to export data from DynamoDB, as shown in the following screen.

Review the template and complete the missing fields. You start by choosing the schedule and frequency by which you want your data export operation to run.

**To configure the schedule**

- On the pipeline page, click **Schedules**.
  a. In the **DefaultSchedule1** section, set **Name** to **ExportSchedule**.
  b. Set **Period** to **1 Hours**.
  c. Set **Start Date Time** using the calendar to the current date, such as **2012-12-18** and the time to **00:00:00 UTC**.
  d. In **Add an optional field**, select **End Date Time**.
  e. Set **End Date Time** using the calendar to the following day, such as **2012-12-19** and the time to **00:00:00 UTC**.
Important
When a schedule’s startDateTime is in the past, AWS Data Pipeline will backfill your pipeline and begin scheduling runs immediately beginning at startDateTime. For more information, see Schedule (p. 292).

Complete the Fields

Templates are pre-populated with the commonly required fields and you complete the rest. Review the fields and values described in this section and provide any missing information.

Contents
- DataNodes (p. 87)
- EmrCluster (p. 87)
- EmrActivity (p. 88)
- Notifications (p. 89)

DataNodes

Next, complete the data node objects in your pipeline definition template.

To configure the DynamoDB data node

1. On the pipeline page, select DataNodes.
2. In the DataNodes pane, in Table Name, type the name of the DynamoDB table that you created in part one of this tutorial; for example: MyTable.
   
   See DynamoDBDataNode (p. 174) for a complete list of fields.

To configure the Amazon S3 data node

1. In the MyS3Data section, in Directory Path, type the path to the files where you want the DynamoDB table data to be written, which is the Amazon S3 bucket that you configured in part one of this tutorial. For example: s3://mybucket/output/MyTable.
   
   See S3DataNode (p. 187) for a complete list of fields.

EmrCluster

Next, complete the resources that will run the data import activities. Many of the fields are auto-populated by the template, as shown in the following screen. You only need to complete the empty fields.
To configure the resources

- On the pipeline page, select Resources.
  - In Emr Log Uri, type the path where to store EMR debugging logs, using the Amazon S3 bucket that you configured in part one of this tutorial; for example: `s3://mybucket/emr_debug_logs`.

See EmrCluster (p. 250) for a complete list of fields.

**EmrActivity**

Next, complete the activity that represents the steps to perform in your data export operation.

To configure the activity

1. On the pipeline page, select Activities.
2. In the MyExportJob section, review the default options already provided. You are not required to manually configure any options in this section.

  **Note**
  The endpoint for your DynamoDB table can be changed by modifying the region value inside the EmrActivity step field. For more information, see EmrActivity (p. 201).

  **Note**
  Consider updating `myDynamoDBReadThroughputRatio`. It sets the rate of read operations to keep your DynamoDB provisioned throughput rate in the allocated range for your table. The value is between 0.1 and 1.5, inclusively. For more information, see Hive Options in the Amazon EMR Developer Guide.

For more information about how to identify which Amazon EMR cluster serves your pipeline, see Identifying the Amazon EMR Cluster that Serves Your Pipeline (p. 154).
Notifications

Next, configure the SNS notification action AWS Data Pipeline must perform depending on the outcome of the activity.

To configure the SNS success, failure, and late notification action

1. On the pipeline page, in the right pane, click Others.
2. In the Others pane:
   a. In the LateSnsAlarm section, in Topic Arn, enter the ARN of your Amazon SNS topic that you created earlier in the tutorial; for example: `arn:aws:sns:us-east-1:403EXAMPLE:my-example-topic`.
   b. In the FailureSnsAlarm section, in Topic Arn, enter the ARN of your Amazon SNS topic that you created earlier in the tutorial; for example: `arn:aws:sns:us-east-1:403EXAMPLE:my-example-topic`.
   c. In the SuccessSnsAlarm section, in Topic Arn, enter the ARN of your Amazon SNS topic that you created earlier in the tutorial; for example: `arn:aws:sns:us-east-1:403EXAMPLE:my-example-topic`.

See SnsAlarm (p. 289) for a complete list of fields.

Validate and Save 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 validate and save your pipeline

1. On the pipeline page, click 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, click Close and then, in the right pane, click Errors/Warnings.
3. The Errors/Warnings pane lists the objects that failed validation. Click 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, click 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. On the List Pipelines page, in the Details column of your pipeline, click View pipeline.
2. In the pipeline page, click Activate.
3. In the confirmation dialog box, click Close.

Monitor the Progress of Your Pipeline Runs

You can monitor the progress of your pipeline. For more information about instance status, see Interpreting Pipeline Status Details (p. 155). For more information about troubleshooting failed or incomplete instance runs of your pipeline, see Resolving Common Problems (p. 156).

To monitor the progress of your pipeline using the console

1. On the List Pipelines page, in the Details column for your pipeline, click View instance details.
2. The Instance details page lists the status of each instance.
   
   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 instances 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 instances 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.

   a. Click the triangle next to an instance.
   b. In the Instance summary pane, click View instance fields to see the fields associated with the selected instance. If the status of the instance is FAILED, the details box has an entry indicating the reason for failure. For example, @failureReason = Resource not healthy terminated.
   c. In the Instance summary pane, in the Select attempt for this instance box, select the attempt number.
   d. In the Instance summary pane, click View attempt fields to see details of fields associated with the selected attempt.

5. To take an action on your incomplete or failed instance, select an action (Rerun|Cancel|Mark Finished) from the Action column of the instance.
Verify 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-fbfea01a371e_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, ProductCategory, as shown in the following screen. The presence of this text file indicates that the export operation from DynamoDB to the output file occurred successfully.

Note
The control-character delimited text file uses the Start of Text (STX/ASCII 02) and End of Text (ETX/ASCII 03) characters to indicate the beginning and end of the data fields/columns, respectively. A single line feed (LF/ASCII 10) indicates the end of records.

(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 using the console

1. In the List Pipelines page, click the check box next to your pipeline.
2. Click Delete.
3. In the confirmation dialog box, click Delete to confirm the delete request.

Copy CSV Data from Amazon S3 to Amazon S3

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.

This tutorial uses the following objects to create a pipeline definition:

Activity
Activity the AWS Data Pipeline performs for this pipeline.
This tutorial uses the CopyActivity object to copy CSV data from one Amazon S3 bucket to another.

**Important**
There are distinct limitations regarding the CSV file format with CopyActivity and S3DataNode. For more information, see CopyActivity (p. 196).

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

**Resource**
Resource AWS Data Pipeline must use to perform this activity.

This tutorial uses Ec2Resource, an EC2 instance provided by AWS Data Pipeline, to copy data. AWS Data Pipeline automatically launches the EC2 instance and then terminates the instance after the task finishes.

**DataNodes**
Input and output nodes for this pipeline.

This tutorial uses S3DataNode for both input and output nodes.

**Action**
Action AWS Data Pipeline must take when the specified conditions are met.

This tutorial uses SnsAlarm action to send Amazon SNS notifications to the Amazon SNS topic you specify, after the task finishes successfully. You must subscribe to the Amazon SNS Topic Arn to receive the notifications.

---

**Before You Begin**

Be sure you've completed the following steps.

- Complete the tasks in Setting Up AWS Data Pipeline (p. ?).
- (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. 46).
- 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 an Amazon SNS 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 Setting Up IAM Roles (p. 4).

**Note**
Some of the actions described in this tutorial can generate AWS usage charges, depending on whether you are using the AWS Free Usage Tier.
Using the AWS Data Pipeline Console

To create the pipeline using the AWS Data Pipeline console, complete the following tasks.

Tasks

- Create and Configure the Pipeline Definition (p. 93)
- Validate and Save Your Pipeline (p. 96)
- Verify your Pipeline Definition (p. 96)
- Activate your Pipeline (p. 96)
- Monitor the Progress of Your Pipeline Runs (p. 97)
- (Optional) Delete your Pipeline (p. 98)

Create and Configure the Pipeline Definition

First, create the pipeline definition.

To create your pipeline definition

1. Open the AWS Data Pipeline console.
2. Click either Create new pipeline or Get started now (if you haven't created a pipeline in this region).
3. On the Create Pipeline page:
   a. In the Name field, enter a name (for example, CopyMyS3Data).
   b. In Description, enter a description.
   c. Choose whether to run the pipeline once on activation or on a schedule.
   d. Leave IAM roles set to its default value, which is to use the default IAM roles, DataPipelineDefaultRole for the pipeline role and DataPipelineDefaultResourceRole for the resource role.
   Note
      If you have created your own custom IAM roles and would like to use them in this tutorial, you can select them now.
   e. Click Create.

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

1. On the pipeline page, select Add activity.
2. In the Activities pane:
   a. In the Name field, enter a name for the activity, for example, copy-myS3-data.
   b. In the Type field, select CopyActivity.
   c. In the Schedule field, select Create new: Schedule.
   d. In the Input field, select Create new: DataNode.
   e. In the Output field, select Create new: DataNode.
   f. In the Add an optional field field, select RunsOn.
   g. In the Runs On field, select Create new: Resource.
h. In the **Add an optional field** field, select **On Success**.
i. In the **On Success** field, select **Create new: Action**.
j. In the left pane, separate the icons by dragging them apart.

You've completed defining your pipeline definition by specifying the objects AWS Data Pipeline uses to perform the copy activity.

The **Pipeline** pane shows the graphical representation of the pipeline you just created. The arrows indicate the connection between the various objects.

Next, configure the run date and time for your pipeline.

**To configure run date and time for your pipeline**

1. On the pipeline page, in the right pane, click **Schedules**.
2. In the **Schedules** pane:
   a. Enter a schedule name for this activity (for example, `copy-myS3-data-schedule`).
   b. In the **Start Date Time** field, select the date from the calendar, and then enter the time to start the activity.

   **Note**
   AWS Data Pipeline supports the date and time expressed in "YYYY-MM-DDTHH:MM:SS" format in UTC/GMT only.
   
   c. In the **Period** field, enter the duration for the activity (for example, 1), and then select the period category (for example, *Days*).
   d. (Optional) To specify the date and time to end the activity, in the **Add an optional field** 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.

Next, configure the input and 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. In the **DataNodes** pane:
   a. In the **DefaultDataNode1 Name** field, enter the name for your input node (for example, MyS3Input).
      
      In this tutorial, your input node is the Amazon S3 data source bucket.
   b. In the **Type** field, select **S3DataNode**.
   c. In the **Schedule** field, select **copy-myS3-data-schedule**.
   d. In the **Add an optional field** field, select **File Path**.
   e. In the **File Path** field, enter the path to your Amazon S3 bucket (for example, s3://my-data-pipeline-input/name of your data file).
   f. In the **DefaultDataNode2 Name** field, enter the name for your output node (for example, MyS3Output).
      
      In this tutorial, your output node is the Amazon S3 data target bucket.
   g. In the **Type** field, select **S3DataNode**.
   h. In the **Schedule** field, select **copy-myS3-data-schedule**.
   i. In the **Add an optional field** field, select **File Path**.
   j. In the **File Path** field, enter the path to your Amazon S3 bucket (for example, s3://my-data-pipeline-output/name of your data file).

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

**To configure the resource**

1. On the pipeline page, in the right pane, click **Resources**.
2. In the **Resources** pane:
   a. In the **Name** field, enter the name for your resource (for example, CopyDataInstance).
   b. In the **Type** field, select **Ec2Resource**.
   c. [EC2-VPC] In **Add an optional field**, select **Subnet Id**.
   d. [EC2-VPC] In **Subnet Id**, enter the ID of the subnet.
   e. In the **Schedule** field, select **copy-myS3-data-schedule**.
   f. Leave the **Role** and **Resource Role** field set to default values for this tutorial.
      
      **Note**
      If you have created your own IAM roles, you can select them now.

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

**To configure the Amazon SNS notification action**

1. On the pipeline page, in the right pane, click **Others**.
2. In the **Others** pane:
   a. In the **DefaultAction1 Name** field, enter the name for your Amazon SNS notification (for example, CopyDataNotice).
   b. In the **Type** field, select **SnsAlarm**.
   c. In the **Subject** field, enter the subject line for your notification.
d. Leave the Role field set to the default value for this tutorial.

e. In the Message field, enter the message content.

f. In the Topic Arn field, enter the ARN of your Amazon SNS topic.

You have now completed all the steps required for creating your pipeline definition. Next, validate and save your pipeline.

**Validate and Save 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 validate and save your pipeline

1. On the pipeline page, click **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, click **Close** and then, in the right pane, click **Errors/Warnings**.

3. The Errors/Warnings pane lists the objects that failed validation. Click 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, click **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, check if your newly-created pipeline is listed.

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

   The Status column in the row listing your pipeline should show PENDING.

2. Click on the triangle icon next to your pipeline. The Pipeline summary pane below shows the details of your pipeline runs. Because your pipeline is not yet activated, you should see only 0s at this point.

3. In the Pipeline summary pane, click **View fields** to see the configuration of your pipeline definition.

4. Click **Close**.

**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. On the List Pipelines page, in the Details column of your pipeline, click View pipeline.
2. In the pipeline page, click Activate.
3. In the confirmation dialog box, click Close.

Monitor the Progress of Your Pipeline Runs

You can monitor the progress of your pipeline. For more information about instance status, see Interpreting Pipeline Status Details (p. 155). For more information about troubleshooting failed or incomplete instance runs of your pipeline, see Resolving Common Problems (p. 156).

To monitor the progress of your pipeline using the console
1. On the List Pipelines page, in the Details column for your pipeline, click View instance details.
2. The Instance details page lists the status of each instance.
   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 instances 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 instances 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.
   a. Click the triangle next to an instance.
   b. In the Instance summary pane, click View instance fields to see the fields associated with the selected instance. If the status of the instance is FAILED, the details box has an entry indicating the reason for failure. For example, @failureReason = Resource not healthy terminated.
   c. In the Instance summary pane, in the Select attempt for this instance box, select the attempt number.
   d. In the Instance summary pane, click View attempt fields to see details of fields associated with the selected attempt.
5. To take an action on your incomplete or failed instance, select an action (Rerun, Cancel, Mark Finished) from the Action column of the instance.

(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 using the console
1. In the List Pipelines page, click the check box next to your pipeline.
2. Click Delete.
3. In the confirmation dialog box, click Delete to confirm the delete request.

Using the Command Line Interface

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

Prerequisites

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

1. Select, install, and configure a CLI. For more information, see (Optional) Installing a Command Line Interface (p. 3).
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, you must create these roles manually. For more information, see Setting Up IAM Roles (p. 4).

Tasks
• Define a Pipeline in JSON Format (p. 98)
• Upload and Activate the Pipeline Definition (p. 102)

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:

```json
{
  "objects": [
    {
      "id": "MySchedule",
      "type": "Schedule",
      "startDateTime": "2013-08-18T00:00:00"
    }
  ]
}
```
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.
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:

```json
{
  "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.

```json
{
  "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:

```json
{
  "id": "MyEC2Resource",
```
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
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.
**Export MySQL Data to Amazon S3 with Copy-Activity**

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.

This tutorial uses the following pipeline objects:

- CopyActivity (p. 196)
- Ec2Resource (p. 244)
- MySqlDataNode (p. 179)
- S3DataNode (p. 187)
- SnsAlarm (p. 289)

**Before You Begin**

Be sure you’ve completed the following steps.

- Complete the tasks in Setting Up AWS Data Pipeline (p. ?).
- (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. 46).
- 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 will need all this information in this tutorial.

- 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, go to Create a Table in the MySQL documentation and the MySQL Workbench product page.

- Create an Amazon SNS topic for sending email notification and make a note of the topic Amazon Resource Name (ARN). For more information, go to 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 Setting Up IAM Roles (p. 4).

Note
Some of the actions described in this tutorial can generate AWS usage charges, depending on whether you are using the AWS Free Usage Tier.

Using the AWS Data Pipeline Console

To create the pipeline using the AWS Data Pipeline console, complete the following tasks.

Tasks
- Create and Configure the Pipeline Definition (p. 104)
- Validate and Save Your Pipeline (p. 107)
- Verify Your Pipeline Definition (p. 107)
- Activate your Pipeline (p. 108)
- Monitor the Progress of Your Pipeline Runs (p. 108)
- (Optional) Delete your Pipeline (p. 109)

Create and Configure the Pipeline Definition

First, create the pipeline definition.
To create your pipeline definition

1. Open the AWS Data Pipeline console.
2. Click either Create new pipeline or Get started now (if you haven't created a pipeline in this region).
3. On the Create Pipeline page:
   a. In the Name field, enter a name (for example, CopyMySQLData).
   b. In Description, enter a description.
   c. Choose whether to run the pipeline once on activation or on a schedule.
   d. Leave IAM roles set to its default value, which is to use the default IAM roles, DataPipelineDefaultRole for the pipeline role and DataPipelineDefaultResourceRole for the resource role.
   
   **Note**
   If you have created your own custom IAM roles and would like to use them in this tutorial, you can select them now.
   
   e. Click Create.

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

1. On the pipeline page, click Templates and choose the Copy RDS to S3 template. The pipeline will populate with several objects, requiring you to complete the missing fields.

The pipeline pane shows the graphical representation of the pipeline you just created. The arrows indicate the connection between the various objects. You should see a pipeline that looks similar to this:

2. In the Activities pane
   a. In the Add an optional field field, select On Success.
   b. In the On Success field, select Create new: Action.
Next step, configure run date and time for your pipeline.

To configure run date and time for your pipeline

1. On the pipeline page, in the right pane, click Schedules.
2. In the Schedules pane:
   a. In the Start Date Time field, select the date from the calendar, and then enter the time to start the activity.

      Note
      AWS Data Pipeline supports the date and time expressed in "YYYY-MM-DDTHH:MM:SS" UTC format only.

   b. In the Period field, enter the duration for the activity (for example, 1), and then select the period category (for example, Days).
   c. (Optional) To specify the date and time to end the activity, in the Add an optional field field, select endDateTime, 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 will then starting 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.

Next step, 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. In the DataNodes pane:
   a. In the My RDS Data Connection String field, enter the end point of your MySQL database instance in the format: jdbc:mysql://your-sql-instance-name.id.region-name.rds.amazonaws.com:3306/database_name).

      To locate the endpoint details for your Amazon RDS instance, see Connecting to a DB Instance in the Amazon Relational Database Service User Guide.
   b. In the Username field, enter the user name you used when you created your MySQL database instance.
   c. In the *Password field, enter the password you used when you created your MySQL database instance.
   d. In the Table field, enter the name of the source MySQL database table (for example, table-name)
   e. In the Add an optional field field, select Select Query.
   f. In the Select Query field, enter a SQL query for the data to copy. For example, select * from #{table}.

      Note
      The #{table} expression re-uses the table name provided by the Table field. For more information, see Pipeline Expressions and Functions (p. 161).
   g. In the My S3 Data Add an optional field field, select File Path.
   h. In the File Path field, enter the path to your Amazon S3 bucket (for example, s3://your-bucket-name/your-output-folder/output_file.csv).
Next step, configure the Amazon SNS notification action AWS Data Pipeline must perform after the copy activity finishes successfully.

**To configure the Amazon SNS notification action**

1. On the pipeline page, in the right pane, click **Others**.
2. In the **Others** pane:
   a. **Note**
      The template pre-configures My Failure Alarm and only requires a Topic Arn value.
      In the **My Failure Alarm Topic Arn** field, enter the ARN of your Amazon SNS topic.
   b. In the **DefaultAction1 Name** field, enter the name for your Amazon SNS notification (for example, My Success Alarm).
   c. In the **Type** field, select **SnsAlarm**.
   d. In the **Topic Arn** field, enter the ARN of your Amazon SNS topic.
   e. Leave the entry in the **Role** field set to default value.
   f. In the **Subject** field, enter the subject line for your notification (for example, RDS to S3 copy succeeded!).
   g. In the **Message** field, enter the message content.

You have now completed all the steps required for creating your pipeline definition. Next step, validate and save your pipeline.

**Validate and Save 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 validate and save your pipeline**

1. On the pipeline page, click **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, click **Close** and then, in the right pane, click **Errors/Warnings**.
3. The **Errors/Warnings** pane lists the objects that failed validation. Click 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, click **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, check if your newly-created pipeline is listed.
   
   AWS Data Pipeline has created a unique Pipeline ID for your pipeline definition.
   
   The Status column in the row listing your pipeline should show PENDING.
2. Click on the triangle icon next to your pipeline. The Pipeline summary pane below shows the details of your pipeline runs. Because your pipeline is not yet activated, you should see only 0s at this point.
3. In the Pipeline summary pane, click View fields to see the configuration of your pipeline definition.
4. Click Close.

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. On the List Pipelines page, in the Details column of your pipeline, click View pipeline.
2. In the pipeline page, click Activate.
3. In the confirmation dialog box, click Close.

Monitor the Progress of Your Pipeline Runs

You can monitor the progress of your pipeline. For more information about instance status, see Interpreting Pipeline Status Details (p. 155). For more information about troubleshooting failed or incomplete instance runs of your pipeline, see Resolving Common Problems (p. 156).

To monitor the progress of your pipeline using the console

1. On the List Pipelines page, in the Details column for your pipeline, click View instance details.
2. The Instance details page lists the status of each instance.
   
   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 instances 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 instances 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.
   
   a. Click the triangle next to an instance.
   b. In the Instance summary pane, click View instance fields to see the fields associated with the selected instance. If the status of the instance is FAILED, the details box has an entry indic-
ating the reason for failure. For example, @failureReason = Resource not healthy terminated.

c. In the **Instance summary** pane, in the **Select attempt for this instance** box, select the attempt number.

d. In the **Instance summary** pane, click **View attempt fields** to see details of fields associated with the selected attempt.

![Instance summary pane](image)

5. To take an action on your incomplete or failed instance, select an action (Rerun|Cancel|Mark Finished) from the **Action** column of the instance.

**(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 using the console**

1. In the **List Pipelines** page, click the check box next to your pipeline.
2. Click **Delete**.
3. In the confirmation dialog box, click **Delete** to confirm the delete request.

**Using the Command Line Interface**

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

**Prerequisites**

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

1. Select, install, and configure a CLI. For more information, see *(Optional) Installing a Command Line Interface* (p. 3).
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, you must create these roles manually. For more information, see **Setting Up IAM Roles** (p. 4).
3. Set up an Amazon S3 bucket and an Amazon RDS instance. For more information, see **Before You Begin** (p. 103).

**Tasks**
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.

```json
{
  "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"
    }
  ]
}
```
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
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. 171).

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:

```json
{
    "id": "S3DataNodeId114",
    "schedule": {
        "ref": "ScheduleId113"
    },
    "name": "S3 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}"
}
```
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:

```json
{
  "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. 289) to send if the activity completes successfully

onFail
The SnsAlarm (p. 289) 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.

```bash
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.

```bash
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.

```bash
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.

```bash
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.

```bash
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. 153).

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

```bash
datapipeline --list-pipelines
```
Copying DynamoDB Data Across Regions

This tutorial walks you through the process of using the CrossRegion DynamoDB Copy template in the AWS Data Pipeline console or using the CLI manually to create a pipeline that can periodically move data between DynamoDB tables across regions or to a different table within the same region.

This feature is useful in the following scenarios:

• Disaster recovery in the case of data loss or region failure
• Moving DynamoDB data across regions to support applications in those regions
• Performing full or incremental DynamoDB data backups

You can use this template to perform DynamoDB backups if you make full table copies each time. For incremental backups, create a lastUpdatedTimestamp attribute and a logical delete attribute (such as IsDeleted) to mark items. Using these two, you can achieve incremental synchronization functionality between two DynamoDB tables.

**Important**
This proposed configuration is a scheduled copy (a snapshot) and not continuous data replication. As a result, if the primary DynamoDB table loses data, there can be data loss when restoring from the backup. In addition, this configuration requires the DynamoDB tables to be on the same AWS account.

The following diagram shows how this template copies data from a DynamoDB table in one region to an empty DynamoDB table in a different region. In the diagram, note that the destination table must already exist, with a primary key that matches the source table.
Before You Begin

This tutorial has the following prerequisites.

- Complete the tasks in Setting Up AWS Data Pipeline (p. ?).
- Two existing DynamoDB tables: a source table populated with data in one region, and an empty DynamoDB table in a different region. For instructions to create these tables, see (Prerequisite) Create the DynamoDB Tables (p. 117).
- (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 Setting Up IAM Roles (p. 4).

Note
Some of the actions described in this tutorial can generate AWS usage charges, depending on whether you are using the AWS Free Usage Tier.

(Prerequisite) Create the DynamoDB Tables

Note
You can skip this section if you already have an DynamoDB source table in one region that is pre-populated with data and an empty DynamoDB destination table in a different region. If so, you can proceed to Using the AWS Data Pipeline Console (p. 119).
This section explains how to create the DynamoDB tables that are a prerequisite for this tutorial. For more information, see Working with Tables in DynamoDB in the DynamoDB Developer Guide.

For this tutorial, you need two DynamoDB tables, each in a different AWS region. Use the following procedure to create a source DynamoDB table, then repeat the steps to create a destination DynamoDB table.

To create a DynamoDB table

1. Open the DynamoDB console.
2. From the region list, choose a region.
   
   **Note**
   
   If you are repeating these steps to create your destination DynamoDB table, choose a different region than your source table.

3. Click **Create Table**.
4. On the **Create Table / Primary Key** page, enter a name (for example, MyTable) in the **Table Name** field.
   
   **Note**
   
   Your table name must be unique.

5. In the **Primary Key** section, for the **Primary Key Type** radio button, select **Hash**.
6. In the **Hash Attribute Name** field, select **Number** and enter the string **Id**.
7. Click **Continue**.
8. On the **Create Table / Provisioned Throughput Capacity** page, in the **Read Capacity Units** field, enter 5.
9. In the **Write Capacity Units** field, enter 5.
   
   **Note**
   
   In this example, we use read and write capacity unit values of five because the sample input data is small. You may need a larger value depending on the size of your actual input data set. For more information, see Provisioned Throughput in Amazon DynamoDB in the Amazon DynamoDB Developer Guide.

10. Click **Continue**.
11. On the **Create Table / Throughput Alarms** page, in the **Send notification to** field, enter your email address.

12. If you do not already have source data, populate your source DynamoDB table with sample data. For example purposes, the following screen shows a DynamoDB source table that has been populated with data from a sample, fictional product catalog that we previously imported as part of a different AWS Data Pipeline tutorial. For more information, see Start Import from the DynamoDB Console (p. 77).
13. At this point, your DynamoDB source table is complete. Repeat the previous steps to create your DynamoDB destination table; however, skip the step where you populate the table with sample data.

**Using the AWS Data Pipeline Console**

To create the pipeline using the AWS Data Pipeline console, complete the following tasks.

**Prerequisites**

Before you can start this tutorial, you must have an DynamoDB source table in one region that is pre-populated with data and an empty DynamoDB destination table in a different region. For more information, see Before You Begin (p. 117).

**Tasks**

- Choose the Template and Configure the Fields (p. 119)
- Confirm Your Settings (p. 123)
- Validate and Save Your Pipeline (p. 123)
- Activate your Pipeline (p. 124)
- Monitor the Progress of Your Pipeline Runs (p. 124)
- (Optional) Delete your Pipeline (p. 125)

**Choose the Template and Configure the Fields**

Create the pipeline, choose a template, and complete the fields for the operation that you want to perform. AWS Data Pipeline uses the information you provide in the template to configure the pipeline objects for you.

**To create a pipeline using the CrossRegion DynamoDB Copy template**

1. Open the AWS Data Pipeline console.
2. Click either Create new pipeline or Get started now (if you haven't created a pipeline in this region).
3. On the Create Pipeline page:
   a. In the Name field, enter a name (for example, CrossRegionDynamoDBCopy).
   b. In the Description field, enter a description.
c. Choose whether to run the pipeline once on activation or on a schedule.

d. Leave IAM roles set to its default value, which is to use the default IAM roles, DataPipelineDefaultRole for the pipeline role and DataPipelineDefaultResourceRole for the resource role.

e. Click Create.

4. On the pipeline page, click Templates and choose the CrossRegion DynamoDB Copy template.

5. Next, complete the missing fields in the configuration screen.

   DynamoDB Cross-Region Copy

   DynamoDB cross-region copy allows you to transfer data across DynamoDB tables in different regions.
   You can perform an incremental copy or a full copy.

   **Table Properties**

   Enter the table names to copy between regions:

   **From table**

   **From region**

   **Read % Allocation**

   **To table**

   **To region**

   **Write % Allocation**

   **Schedule**

   Enter a schedule for the transfer:

   **Period**

   **Hour (s)**

   **Start (UTC)**

   **End (UTC)**

   **Data Filtering**

   Filters determine which items and Attributes to copy. Disable the filter to copy all. Specify multiple Attributes with a comma: (time BIGINT, data STRING).

   **Apply filtering**

   **No**

   **Yes**

   **Attributes**

   **Item Filter SQL**

   **Next**

   a. Enter the table names for the source and destination, along with their respective regions.

   b. Set the Read and Write Percentage Allocation as the percentage of total IOPS allocated to this copy. Read and Write Percentage Allocation sets the rate of read and write operations to keep your DynamoDB provisioned throughput rate in the allocated range for your table. The values are a double between .1 and 1.0, inclusively. For more information, see Specifying Read and Write Requirements for Tables in the DynamoDB Developer Guide.

   c. Choose the frequency (period) of this copy, the start time for the first copy, and optionally the end time.

**Full Copy vs. Incremental Copy**

To copy the entire contents of the table (full table copy), do not provide Data Filtering parameters.

**Important**

Because this is a full table copy, deleted items in the source table are not deleted in the destination table as shown in the following diagram.
To ensure that deletes occur correctly, use a logical delete attribute (such as IsDeleted) to mark items, instead of a physical delete (deleting the item completely from the Amazon DynamoDB source table) as shown in the following diagram.
To perform an incremental copy, provide a time stamp value for the Filter SQL field.

To copy a subset of item attributes, configure the attributes filter as a comma-separated list of attribute names.

**Data Backups**

You can use this template to perform DynamoDB backups if you make full table copies each time. For incremental backups, create a lastUpdatedTimestamp attribute and a logical delete attribute (such as IsDeleted) to mark items. Using these two attributes, you can achieve incremental synchronization functionality between two DynamoDB tables.

**Important**

This proposed configuration is a scheduled copy (a snapshot) and not continuous data replication. As a result, if the primary DynamoDB table loses data, there can be data loss when restoring from the backup.
Confirm Your Settings

To confirm your settings

1. After you complete the form, a confirmation dialog appears to summarize how AWS Data Pipeline will perform the copy. AWS Data Pipeline uses Amazon EMR to perform a parallel copy of data direct from one DynamoDB table to the other, with no intermediate staging.

   ![DynamoDB Cross-Region Copy](image)

2. Click **Continue** to complete the pipeline configuration.
3. Next, you should see a pipeline similar to the following:

   ![Pipeline: DynamoDBCopy](image)

Validate and Save 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 validate and save your pipeline

1. On the pipeline page, click **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, click **Close** and then, in the right pane, click **Errors/Warnings**.
3. The **Errors/Warnings** pane lists the objects that failed validation. Click 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, click 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. On the List Pipelines page, in the Details column of your pipeline, click View pipeline.
2. In the pipeline page, click Activate.
3. In the confirmation dialog box, click Close.

**Monitor the Progress of Your Pipeline Runs**

You can monitor the progress of your pipeline. For more information about instance status, see Interpreting Pipeline Status Details (p. 155). For more information about troubleshooting failed or incomplete instance runs of your pipeline, see Resolving Common Problems (p. 156).

**To monitor the progress of your pipeline using the console**

1. On the List Pipelines page, in the Details column for your pipeline, click View instance details.
2. The Instance details page lists the status of each instance.
   - 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 instances 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 instances 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.
   a. Click the triangle next to an instance.
   b. In the Instance summary pane, click View instance fields to see the fields associated with the selected instance. If the status of the instance is FAILED, the details box has an entry indicating the reason for failure. For example, @failureReason = Resource not healthy terminated.
   c. In the Instance summary pane, in the Select attempt for this instance box, select the attempt number.
   d. In the Instance summary pane, click View attempt fields to see details of fields associated with the selected attempt.
5. To take an action on your incomplete or failed instance, select an action (Rerun | Cancel | Mark Finished) from the Action column of the instance.

(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 using the console

1. In the List Pipelines page, click the check box next to your pipeline.
2. Click Delete.
3. In the confirmation dialog box, click Delete to confirm the delete request.

Using the Command Line Interface

You can use the CLI to create a pipeline that can periodically move data between DynamoDB tables across regions or to a different table within the same region.

Prerequisites

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

1. Select, install, and configure a CLI. For more information, see (Optional) Installing a Command Line Interface (p. 3).
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, you must create these roles manually. For more information, see Setting Up IAM Roles (p. 4).
3. Set up an DynamoDB source table in one region that is pre-populated with data and an empty DynamoDB destination table in a different region. For more information, see Before You Begin (p. 117).

Tasks

1. Define a Pipeline in JSON Format (p. 126)
2. Upload and Activate the Pipeline Definition (p. 130)
Define a Pipeline in JSON Format

This example scenario shows how to use JSON pipeline definitions and the AWS Data Pipeline CLI to create a pipeline that can periodically move data between DynamoDB instances across different regions at a specified time interval or to a different table within the same region.

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": "ScheduleId1",
      "startDateTime": "2013-09-02T05:30:00",
      "name": "DefaultSchedule1",
      "type": "Schedule",
      "period": "12 Hours",
      "endDateTime": "2013-09-03T17:30:00"
    },
    {
      "id": "DynamoDBExportDataFormatId4",
      "name": "DefaultDynamoDBExportDataFormat2",
      "type": "DynamoDBExportDataFormat",
      "column": "name datatype"
    },
    {
      "id": "EmrClusterId6",
      "schedule": {
        "ref": "ScheduleId1"
      },
      "masterInstanceType": "m1.large",
      "coreInstanceType": "m1.large",
      "coreInstanceCount": "3",
      "name": "DefaultEmrCluster1",
      "type": "EmrCluster"
    },
    {
      "id": "DynamoDBExportDataFormatId2",
      "name": "DefaultDynamoDBExportDataFormat1",
      "type": "DynamoDBExportDataFormat",
      "column": "name datatype"
    },
    {
      "id": "DynamoDBDataNodeId5",
      "region": "US_WEST_2",
      "schedule": {
        "ref": "ScheduleId1"
      },
      "writeThroughputPercent": "0.3",
      "tableName": "example-table-west",
      "name": "DefaultDynamoDBDataNode2",
      "dataFormat": {
        "ref": "DynamoDBExportDataFormatId4"
      },
      "type": "DynamoDBDataNode"
  ]
}
```
DynamoDB Data Nodes

The input DynamoDBDataNode pipeline component defines a location for the input data; in this case, an DynamoDB table pre-populated with sample data. The input DynamoDBDataNode component is defined by the following fields:

```json
[
  {
    "id": "Default",
    "scheduleType": "timeseries",
    "failureAndRerunMode": "CASCADE",
    "name": "Default",
    "role": "DataPipelineDefaultRole",
    "resourceRole": "DataPipelineDefaultResourceRole"
  },
  {
    "region": "us-east-1",
    "id": "DynamoDBDataNodeId3",
    "schedule": {
      "ref": "ScheduleId1"
    },
    "tableName": "example-table-east",
    "name": "DefaultDynamoDBDataNode1",
    "dataFormat": {
      "ref": "DynamoDBExportDataFormatId2"
    },
    "type": "DynamoDBDataNode",
    "readThroughputPercent": "0.3"
  }
]
```
The output DynamoDBDataNode pipeline component defines a location for the output data; in this case, an empty DynamoDB table with only a primary key defined. The output DynamoDBDataNode component is defined by the following fields:

```
{
  "id": "DefaultDynamoDBDataNode5",
  "region": "US_WEST_2",
  "schedule": {
    "ref": "ScheduleId1"
  },
  "writeThroughputPercent": "0.3",
  "tableName": "example-table-west",
  "name": "DefaultDynamoDBDataNode2",
  "dataFormat": {
    "ref": "DynamoDBExportDataFormatId4"
  },
  "type": "DynamoDBDataNode"
}
```

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

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

Region
The AWS region in which the DynamoDB table exists. For more information, see Regions and Endpoints in the General Reference.

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

tableName
The name of the DynamoDB table. For more information, see Working with Tables in DynamoDB in the DynamoDB Developer Guide.

dataFormat
The format of the data for the HiveCopyActivity to process. For more information, see DynamoDBExportDataFormat (p. 284)

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. For more information, see Specifying Read and Write Requirements for Tables in the DynamoDB Developer Guide.
Schedule
A reference to the schedule component that we created in the preceding lines of the JSON file.

tableName
The name of the DynamoDB table. For more information, see Working with Tables in DynamoDB in the DynamoDB Developer Guide.

dataFormat
The format of the data for the HiveCopyActivity to process. For more information, see DynamoDBExportDataFormat (p. 284).

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. For more information, see Specifying Read and Write Requirements for Tables in the DynamoDB Developer Guide.

Resource
This is a definition of the computational resource that performs the copy operation. In this example, AWS Data Pipeline should automatically create an Amazon EMR cluster to perform the copy task and terminate the resource after the task completes. The fields defined here control the creation and function of the Amazon EMR cluster that does the work. For more information, see EmrCluster (p. 250). The EmrCluster is defined by the following fields:

```json
{
    "id": "EmrClusterId6",
    "schedule": {
        "ref": "ScheduleId1"
    },
    "masterInstanceType": "m1.large",
    "coreInstanceType": "m1.large",
    "coreInstanceCount": "3",
    "subnetId": "subnet-xxxxxxxx"
    "name": "DefaultEmrCluster1",
    "type": "EmrCluster"
}
```

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

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

Schedule
The schedule on which to create this computational resource.

masterInstanceType
The type of Amazon EC2 instance to use for the master node. The default value is m1.small.

coreInstanceType
The type of Amazon EC2 instance to use for core nodes. The default value is m1.small.

coreInstanceCount
The number of core nodes to use for the cluster. The default value is 1.

subnetId
[EC2-VPC] The ID of the subnet to launch the cluster into.

Type
The type of computational resource to perform work; in this case, an Amazon EMR cluster.
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 HiveCopyActivity component to copy data from one DynamoDB table to another. For more information, see HiveCopyActivity (p. 212). The HiveCopyActivity component is defined by the following fields:

```json
{
    "id": "HiveCopyActivityId7",
    "schedule": { "ref": "ScheduleId1" },
    "input": { "ref": "DynamoDBDataNodeId3" },
    "name": "DefaultHiveCopyActivity1",
    "runsOn": { "ref": "EmrClusterId6" },
    "type": "HiveCopyActivity",
    "output": { "ref": "DynamoDBDataNodeId5" }
}
```

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

Input
A reference to the DynamoDB source table.

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 EmrCluster defined previously. Using the runsOn field causes AWS Data Pipeline to create the Amazon EMR cluster for you. The runsOn field indicates that the resource exists in the AWS infrastructure, while a workerGroup value indicates that you want to use your own on-premises resources to perform the work.

Output
A reference to the DynamoDB destination table.

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. 153).

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.
Note
Some of the actions described in this tutorial can generate AWS usage charges, depending on whether you are using the AWS Free Usage Tier.

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.

Tutorials
• Copy Data to Amazon Redshift Using the AWS Data Pipeline Console (p. 133)
• Copy Data to Amazon Redshift Using the AWS Data Pipeline CLI (p. 137)

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 AWS Data Pipeline (p. ?).
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. Select the new cluster security group.
   f. Select EC2 Security Group from Connection Type and the security group that you created in the second step from EC2 Security Group Name.
   g. 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. 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.
   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.

f. Review the specifications for your cluster, and then click Launch Cluster.

Copy Data to Amazon Redshift Using the AWS Data Pipeline Console

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.

Complete the following steps to create the pipeline using the AWS Data Pipeline console.

Note
Before you start this tutorial, complete the prerequisites described in Before You Begin (p. 132).

Tasks
• Choose the Template and Configure the Fields (p. 133)
• Validate and Save Your Pipeline (p. 135)
• Activate your Pipeline (p. 135)
• Monitor the Progress of Your Pipeline Runs (p. 136)
• (Optional) Delete your Pipeline (p. 137)

Choose the Template and Configure the Fields

Note
Before you start this tutorial, complete the prerequisites described in Before You Begin (p. 132).

Choose a template and complete the fields for the operation that you want to perform. AWS Data Pipeline uses the information you provide in the template to configure pipeline objects for you.
To create a pipeline using the Copy to Redshift template

1. Open the AWS Data Pipeline console.
2. Click either Create new pipeline or Get started now (if you haven't created a pipeline in this region).
3. On the Create Pipeline page, enter a name and description for the pipeline. Leave the other settings at their defaults and click Create.
4. On the pipeline page, click Templates and select the Copy to Redshift template.
5. Complete the missing fields in the first configuration screen, and then click Next.
   a. Enter the details for the Amazon Redshift database that you selected or created.
   b. Set the Cluster Identifier to 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 clusterId value is mydb. In the Amazon Redshift console, this value is "Cluster Name".
   c. In Table Name, specify the name of a table for the output.
   d. Select no to indicate that the specified table doesn't exist, and then enter the following SQL statement to create the table for the sample data to use in this tutorial.

   ```sql
   create table StructuredLogs (requestBeginTime CHAR(30) PRIMARY KEY DISTKEY SORTKEY, requestEndTime CHAR(30), hostname CHAR(100), requestDate varchar(20));
   ```
   e. Select an insert type (KEEP_EXISTING, OVERWRITE_EXISTING, or TRUNCATE).
      Warning
      TRUNCATE will delete all the data in the table before writing.

6. Complete the missing fields in the second configuration screen, and then click Next.
   a. Select S3 File Path and specify the following sample data file.

   ```text
   s3://datapipeline-us-east-1/samples/hive-ads-samples.csv
   ```
   b. In the Data format list, select csv.
   c. Select a schedule.
   d. In EC2 Security Group, specify the name of the security group for EC2-Classic that you created for this tutorial.
   e. Specify one of your own Amazon S3 buckets to use for the debug logs.
We display a summary screen for your pipeline. Click **Use Template** to complete the pipeline configuration.

**Validate and Save 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 validate and save your pipeline**

1. On the pipeline page, click **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, click **Close** and then, in the right pane, click **Errors/Warnings**.
3. The **Errors/Warnings** pane lists the objects that failed validation. Click 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, click **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. On the List Pipelines page, in the Details column of your pipeline, click View pipeline.
2. In the pipeline page, click Activate.
3. In the confirmation dialog box, click Close.

Monitor the Progress of Your Pipeline Runs

You can monitor the progress of your pipeline. For more information about instance status, see Interpreting Pipeline Status Details (p. 155). For more information about troubleshooting failed or incomplete instance runs of your pipeline, see Resolving Common Problems (p. 156).

To monitor the progress of your pipeline using the console

1. On the List Pipelines page, in the Details column for your pipeline, click View instance details.
2. The Instance details page lists the status of each instance.
   - 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 instances 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 instances 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.
   - a. Click the triangle next to an instance.
   - b. In the Instance summary pane, click View instance fields to see the fields associated with the selected instance. If the status of the instance is FAILED, the details box has an entry indicating the reason for failure. For example, @failureReason = Resource not healthy terminated.
   - c. In the Instance summary pane, in the Select attempt for this instance box, select the attempt number.
   - d. In the Instance summary pane, click View attempt fields to see details of fields associated with the selected attempt.

5. To take an action on your incomplete or failed instance, select an action (Rerun|Cancel|Mark Finished) from the Action column of the instance.
(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 using the console**

1. In the List Pipelines page, click the check box next to your pipeline.
2. Click Delete.
3. In the confirmation dialog box, click Delete to confirm the delete request.

Copy Data to Amazon Redshift Using the AWS Data Pipeline CLI

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

1. Select, install, and configure a CLI. For more information, see *(Optional) Installing a Command Line Interface (p. 3)*.
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, you must create these roles manually. For more information, see **Setting Up IAM Roles (p. 4)**.
3. Set up an Amazon Redshift database. For more information, see **Before You Begin (p. 132)**.

**Tasks**

- **Define a Pipeline in JSON Format (p. 137)**
- **Upload and Activate the Pipeline Definition (p. 143)**

**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.

```json
{
    "objects": [
        {
            "id": "CSVId1",
```

---

API Version 2012-10-29

137
"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",
"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"
}
}
Objects

- Data Nodes (p. 139)
- Resource (p. 141)
- Activity (p. 142)

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. 187).

This input component is defined by the following fields:

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

For more information about these objects, see the following documentation.
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. 183). This output component is defined by the following fields:

```json
{
  "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. 277).

```json
{
    "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. 244).

The Ec2Resource is defined by the following fields:

```json
{
    "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 granu-
    larity 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. 227).

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. 153).

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

```
datapipeline --list-pipelines
```
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 Task Runner on User-Managed Resources (p. 146).

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.

```json
{
    "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",
    "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": "ml.xlarge",
    "coreInstanceType": "ml.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.

## Task Runner on User-Managed Resources

You can install Task Runner on computational resources that you manage, such as a long-running 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 `worker-Group` 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.
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 http://aws.amazon.com/developertools/AWS-Data-Pipeline/1920924250474601 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). Copy the file to the directory where you installed Task Runner. This file has the same structure as the credentials file you use for the AWS Data Pipeline CLI. For more information, see Configure Credentials for the CLI (p. 298).

For CLI access, you need an access key ID and secret access key. Use IAM user access keys instead of AWS root account access keys. IAM lets you securely control access to AWS services and resources in your AWS account. 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
```
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 Pre-conditions (p. 15).

Task Runner Configuration Options

These are the configuration options available from the command line when you launch Task Runner.

<table>
<thead>
<tr>
<th>Command Line Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--help</code></td>
<td>Command line help. Example: <code>Java -jar TaskRunner-1.0.jar --help</code></td>
</tr>
<tr>
<td><code>--config</code></td>
<td>The path and file name of your <code>credentials.json</code> file.</td>
</tr>
<tr>
<td><code>--accessId</code></td>
<td>Your AWS access key ID for Task Runner to use when making requests. The <code>--accessId</code> and <code>--secretKey</code> options provide an alternative to using a credentials.json file. If a <code>credentials.json</code> file is also provided, the <code>--accessId</code> and <code>--secretKey</code> options will take precedence.</td>
</tr>
<tr>
<td><code>--secretKey</code></td>
<td>Your AWS secret key for Task Runner to use when making requests. For more information, see <code>--accessId</code>.</td>
</tr>
<tr>
<td>Command Line Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>--endpoint</td>
<td>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.</td>
</tr>
<tr>
<td>--workerGroup</td>
<td>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.</td>
</tr>
<tr>
<td>--taskrunnerId</td>
<td>The ID of the task runner to use when reporting progress. Optional.</td>
</tr>
<tr>
<td>--output</td>
<td>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.</td>
</tr>
<tr>
<td>--tasks</td>
<td>The number of task poll threads to run simultaneously. Optional. The default is 2.</td>
</tr>
<tr>
<td>--region</td>
<td>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.</td>
</tr>
<tr>
<td>--logUri</td>
<td>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.</td>
</tr>
<tr>
<td>--proxyHost</td>
<td>The host of the proxy which Task Runner clients will use to connect to AWS Services.</td>
</tr>
<tr>
<td>--proxyPort</td>
<td>Port of the proxy host which the Task Runner clients will use to connect to AWS Services.</td>
</tr>
<tr>
<td>--proxyUsername</td>
<td>The username for proxy.</td>
</tr>
<tr>
<td>--proxyPassword</td>
<td>The password for proxy.</td>
</tr>
<tr>
<td>Command Line Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>--proxyDomain</td>
<td>The Windows domain name for NTLM Proxy.</td>
</tr>
<tr>
<td>--proxyWorkstation</td>
<td>The Windows workstation name for NTLM Proxy.</td>
</tr>
</tbody>
</table>

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

AWS Data Pipeline automatically creates Amazon EC2 instances, and installs and configures Task Runner for you when you specify an Ec2Resource object in your pipeline. However, it is possible to create and use a custom AMI with AWS Data Pipeline on which to run Task Runner. Prepare a custom AMI and specify its AMI ID using the imageId field on an Ec2Resource object. For more information, see Ec2Resource (p. 244).

A custom AMI must meet several prerequisites for AWS Data Pipeline to use it successfully for Task Runner. The custom AMI must have the following software installed:

- Linux
- Bash
- wget
- unzip
- Java 1.6 or newer
- cloud-init

In addition, you must create and configure a user account named ec2-user on the custom AMI.

For more information, see Creating Your Own AMIs.
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 identify the problem and find a solution.

Contents
- AWS Data Pipeline Troubleshooting In Action (p. 153)
- Locating Errors in Pipelines (p. 153)
- Identifying the Amazon EMR Cluster that Serves Your Pipeline (p. 154)
- Interpreting Pipeline Status Details (p. 155)
- Locating Error Logs (p. 156)
- Resolving Common Problems (p. 156)

AWS Data Pipeline Troubleshooting In Action

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. Pipelines have a SCHEDULED status if they have passed validation and are ready, currently performing work, or done with their work. PENDING status means the pipeline is not able to perform work for some reason; for example, the pipeline definition might be incomplete or might have failed the validation step that all pipelines go through before activation. 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.

Pipeline components have the following status values:

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

- **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` field defined by that component.

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

- **VALIDATING**
  - The pipeline definition is being validated by AWS Data Pipeline.

- **RUNNING**
  - The resource is running and ready to receive work.

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

- **TIMEOUT**
  - 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.

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

- **FINISHED**
  - The component completed its assigned work.

- **SHUTTING_DOWN**
  - The resource is shutting down after successfully completed its work.

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

- **CASCADE_FAILED**
  - The component or resource was canceled as a result of a cascade failure from one of its dependencies, but was probably not the original source of the failure.
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. 150).

To configure the log location using the AWS Data Pipeline CLI in a pipeline JSON file, begin your pipeline file with the following text:

```json
{
  "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.

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 (p. 157)
- Pipeline Component Stuck in Waiting for Runner Status (p. 157)
- Pipeline Component Stuck in WAITING_ON_DEPENDENCIES Status (p. 157)
- Run Doesn’t Start When Scheduled (p. 158)
- Pipeline Components Run in Wrong Order (p. 158)
- EMR Cluster Fails With Error: The security token included in the request is invalid (p. 159)
- Insufficient Permissions to Access Resources (p. 159)
- Status Code: 400 Error Code: PipelineNotFound异常 (p. 159)
- Creating a Pipeline Causes a Security Token Error (p. 159)
- Cannot See Pipeline Details in the Console (p. 159)
- Error in remote runner Status Code: 404, AWS Service: Amazon S3 (p. 159)
- Access Denied - Not Authorized to Perform Function datapipeline: (p. 159)
- Increasing AWS Data Pipeline Limits (p. 160)
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.

```json
{
  "id": "InputData",
}
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. 18) 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:

```json
{
    "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 Setting Up IAM Roles (p. 4).

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 Setting Up IAM Roles (p. 4).

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 Setting Up IAM Roles (p. 4).

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

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.

```json
{
  "Action": [ "datapipeline:PollForTask" ],
  "Effect": "Allow",
  "Resource": ["*"]
}
```

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 Web Service Limits (p. 322). 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.
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. 161)
- Numeric (p. 161)
- Object References (p. 162)
- Period (p. 162)
- String (p. 162)

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. 163). For more information about the pipeline object types, see Pipeline Object Reference (p. 173).

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 slash 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. 163) and Expression Evaluation (p. 165).

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. 165).
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".

```json
{
  "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.

```json
{
  "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:

```bash
#{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:
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.

```
{
    "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. 54).

```
{
    "name": "DefaultActivity1",
    "type": "CopyActivity",
    "schedule": {"ref": "Once"},
```
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.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition. Example: #{1 + 2} Result: 3</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction. Example: #{1 - 2} Result: -1</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication. Example: #{1 * 2} Result: 2</td>
</tr>
<tr>
<td>/</td>
<td>Division. If you divide two integers, the result is truncated. Example: #{1 / 2}, Result: 0 Example: #{1.0 / 2}, Result: .5</td>
</tr>
</tbody>
</table>
String Functions

The following functions are available for working with string values.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>^</td>
<td>Exponent.</td>
</tr>
<tr>
<td></td>
<td>Example: #{2 ^ 2}</td>
</tr>
<tr>
<td></td>
<td>Result: 4.0</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int day(DateTime myDateTime)</td>
<td>Gets the day of the DateTime value as an integer.</td>
</tr>
<tr>
<td></td>
<td>Example: #{day(myDateTime)}</td>
</tr>
<tr>
<td></td>
<td>Result: 24</td>
</tr>
<tr>
<td>int dayOfYear(DateTime myDateTime)</td>
<td>Gets the day of the year of the DateTime value as an integer.</td>
</tr>
<tr>
<td></td>
<td>Example: #{dayOfYear(myDateTime)}</td>
</tr>
<tr>
<td></td>
<td>Result: 144</td>
</tr>
</tbody>
</table>
# Date and Time Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| `DateTime firstOfMonth(DateTime myDateTime)` | Creates a `DateTime` object for the start of the month in the specified `DateTime`.  
Example:  
```java
#{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:  
```java
#{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:  
```java
#{hour(myDateTime)}
```
Result:  
17

| `DateTime inTimeZone(DateTime myDateTime, String zone)` | Creates a `DateTime` object with the same date and time, but in the specified time zone, and taking daylight savings time into account. For more information about time zones, see [http://joda-time.sourceforge.net/timezones.html](http://joda-time.sourceforge.net/timezones.html).  
Example:  
```java
#{inTimeZone(myDateTime,'America/Los_Angeles')}
```
Result:  
"2011-05-24T10:10:00 America/Los_Angeles"

| `DateTime makeDate(int year, int month, int day)` | Creates a `DateTime` object, in UTC, with the specified year, month, and day, at midnight.  
Example:  
```java
#{makeDate(2011,5,24)}
```
Result:  
"2011-05-24T0:00:00z"
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DateTime makeDateTime(int year, int month, int day, int hour, int minute)</td>
<td>Creates a DateTime object, in UTC, with the specified year, month, day, hour, and minute. &lt;br&gt;Example: <code>#{makeDateTime(2011,5,24,14,21)}</code>&lt;br&gt;Result: &quot;2011-05-24T14:21:00z&quot;</td>
</tr>
<tr>
<td>DateTime midnight(DateTime myDateTime)</td>
<td>Creates a DateTime object for the next midnight, relative to the specified DateTime. &lt;br&gt;Example: <code>#{midnight(myDateTime)}</code>&lt;br&gt;Result: &quot;2011-05-25T0:00:00z&quot;</td>
</tr>
<tr>
<td>DateTime minusDays(DateTime myDateTime, int daysToSub)</td>
<td>Creates a DateTime object that is the result of subtracting the specified number of days from the specified DateTime. &lt;br&gt;Example: <code>#{minusDays(myDateTime,1)}</code>&lt;br&gt;Result: &quot;2011-05-23T17:10:00z&quot;</td>
</tr>
<tr>
<td>DateTime minusHours(DateTime myDateTime, int hoursToSub)</td>
<td>Creates a DateTime object that is the result of subtracting the specified number of hours from the specified DateTime. &lt;br&gt;Example: <code>#{minusHours(myDateTime,1)}</code>&lt;br&gt;Result: &quot;2011-05-24T16:10:00z&quot;</td>
</tr>
<tr>
<td>DateTime minusMinutes(DateTime myDateTime, int minutesToSub)</td>
<td>Creates a DateTime object that is the result of subtracting the specified number of minutes from the specified DateTime. &lt;br&gt;Example: <code>#{minusMinutes(myDateTime,1)}</code>&lt;br&gt;Result: &quot;2011-05-24T17:09:00z&quot;</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| 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 |
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>
| `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

<table>
<thead>
<tr>
<th>DateTime plusYears(DateTime myDateTime, int yearsToAdd)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Creates a DateTime object that is the result of adding the specified number of years to the specified DateTime.</td>
</tr>
<tr>
<td>Example:</td>
</tr>
<tr>
<td>${plusYears(myDateTime,1)}</td>
</tr>
<tr>
<td>Result:</td>
</tr>
<tr>
<td>&quot;2012-05-24T17:10:00z&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DateTime sunday(DateTime myDateTime)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>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.</td>
</tr>
<tr>
<td>Example:</td>
</tr>
<tr>
<td>${sunday(myDateTime)}</td>
</tr>
<tr>
<td>Result: &quot;2011-05-22 17:10:00 UTC&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int year(DateTime myDateTime)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Gets the year of the DateTime value as an integer.</td>
</tr>
<tr>
<td>Example:</td>
</tr>
<tr>
<td>${year(myDateTime)}</td>
</tr>
<tr>
<td>Result: 2011</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DateTime yesterday(DateTime myDateTime)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>Creates a DateTime object for the previous day, relative to the specified DateTime. The result is the same as minusDays(1).</td>
</tr>
<tr>
<td>Example:</td>
</tr>
<tr>
<td>${yesterday(myDateTime)}</td>
</tr>
<tr>
<td>Result: &quot;2011-05-23T17:10:00z&quot;</td>
</tr>
</tbody>
</table>

### Special Characters

AWS Data Pipeline uses certain characters that have a special meaning in pipeline definitions, as shown in the following table.

<table>
<thead>
<tr>
<th>Special Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@</td>
<td>Runtime field. This character is a field name prefix for a field that is only available when a pipeline runs.</td>
</tr>
</tbody>
</table>
### Special Characters

<table>
<thead>
<tr>
<th>Special Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>Expression. Expressions are delimited by: &quot;#{ &quot; and &quot; }&quot; and the contents of the braces are evaluated by AWS Data Pipeline. For more information, see Expressions (p. 162).</td>
</tr>
<tr>
<td>*</td>
<td>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.</td>
</tr>
</tbody>
</table>
Pipeline Object Reference

This section describes the pipeline objects/components that you can use in your pipeline definition file.

Topics

- Object Hierarchy (p. 173)
- DataNodes (p. 174)
- Activities (p. 196)
- Resources (p. 244)
- Preconditions (p. 256)
- Databases (p. 275)
- Data Formats (p. 279)
- Actions (p. 289)
- Schedule (p. 292)

Object Hierarchy

The following is the object hierarchy for AWS Data Pipeline.
DataNodes

The following are Data Pipeline DataNodes:

Topics
- DynamoDBDataNode (p. 174)
- MySqlDataNode (p. 179)
- RedshiftDataNode (p. 183)
- S3DataNode (p. 187)
- SqlDataNode (p. 191)

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.

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

Syntax

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>dynamoDBDataFormat</td>
<td>Applies a schema to a DynamoDB table to make it accessible by a Hive query.</td>
<td>DynamoDBDataFormat</td>
<td>No</td>
</tr>
<tr>
<td>precondition</td>
<td>A list of preconditions to be met. A data node is not marked READY until all preconditions are met.</td>
<td>List</td>
<td>No</td>
</tr>
</tbody>
</table>
### DynamoDBDataNode

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>readThroughputPercent</td>
<td>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. For more information, see Specifying Read and Write Requirements for Tables.</td>
<td>Double</td>
<td>No</td>
</tr>
<tr>
<td>region</td>
<td>The AWS region where the Dynamo table exists. It’s used by HiveActivity when it performs staging for DynamoDB tables in Hive. For more information, see Using a Pipeline with Resources in Multiple Regions (p. 50).</td>
<td>Region string</td>
<td>No</td>
</tr>
<tr>
<td>tableName</td>
<td>The DynamoDB table.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>writeThroughputPercent</td>
<td>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. For more information, see Specifying Read and Write Requirements for Tables.</td>
<td>Double</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields from the `DataNode` object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>onFail</td>
<td>The SnsAlarm to use when the current instance fails.</td>
<td>SnsAlarm (p. 289) object reference</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>The SnsAlarm to use when the current instance succeeds.</td>
<td>SnsAlarm (p. 289) object reference</td>
<td>No</td>
</tr>
<tr>
<td>precondition</td>
<td>A list of precondition objects that must be true for the data node to be valid. A data node cannot reach the READY status until all its conditions are met. Preconditions do not have their own schedule or identity, instead they run on the schedule of the activity or data node with which they are associated.</td>
<td>A list of object references</td>
<td>No</td>
</tr>
<tr>
<td>schedule</td>
<td>A schedule of the object. A common use is to specify a time schedule that correlates to the schedule for the object. This slot overrides the schedule slot included from <code>ScheduleableObject</code>, which is optional.</td>
<td>Schedule (p. 292) object reference</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### scheduleType
- **Description**: Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of the interval or 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.
- **Type**: Allowed values are "cron" or "timeseries". Defaults to "timeseries".
- **Required**: No

### workerGroup
- **Description**: The worker group. This is used for routing tasks. If you provide a `runsOn` value and `workerGroup` exists, `workerGroup` is ignored.
- **Type**: String
- **Required**: No

This object includes the following fields from RunnableObject.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>@activeInstances</td>
<td>Record of the currently scheduled instance objects</td>
<td>Schedulable object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptStatus</td>
<td>The status most recently reported from the object</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@actualEndTime</td>
<td>The date and time that the scheduled run actually ended. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@actualStartTime</td>
<td>The date and time that the scheduled run actually started. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptTimeout</td>
<td>The timeout time interval for an object attempt. If an attempt does not complete within the start time plus this time interval, AWS Data Pipeline marks the attempt as failed and your retry settings determine the next steps taken.</td>
<td>Time period; for example, &quot;1 hour&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@cascadeFailedOn</td>
<td>Description of which dependency the object failed on.</td>
<td>List of objects (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@componentParent</td>
<td>The component from which this instance is created.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorId</td>
<td>If the object failed, the error code. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorMessage</td>
<td>If the object failed, the error message. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorStackTrace</td>
<td>If the object failed, the error stack trace.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>failureAndRerunMode</td>
<td>Determines whether pipeline object failures and rerun commands cascade through pipeline object dependencies. For more information, see Cascading Failures and Reruns (p. 51).</td>
<td>String. Possible values are <code>cascade</code> and <code>none</code>.</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>@headAttempt</td>
<td>The latest attempt on the given instance.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@hostname</td>
<td>The host name of client that picked up the task attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>lateAfterTimeout</td>
<td>The time period in which the object run must start. If the object does not start within the scheduled start time plus this time interval, it is considered late.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>maximumRetries</td>
<td>The maximum number of times to retry the action. The default value is 2, which results in 3 tries total (1 original attempt plus 2 retries). The maximum value is 5 (6 total attempts).</td>
<td>Integer</td>
<td>No</td>
</tr>
<tr>
<td>onFail</td>
<td>An action to run when the current object fails.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onLateAction</td>
<td>The SnsAlarm to use when the object's run is late.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>An action to run when the current object succeeds.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>@reportProgressTime</td>
<td>The last time that Task Runner, or other code that is processing the tasks, called the ReportTaskProgress API.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>reportProgressTimeout</td>
<td>The time period for successive calls from Task Runner to the ReportTaskProgress API. If Task Runner, or other code that is processing the tasks, does not report progress within the specified period, the object attempt can be retried.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@resource</td>
<td>The resource instance on which the given activity/precondition attempt is being run.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>retryDelay</td>
<td>The timeout duration between two retry attempts. The default is 10 minutes.</td>
<td>Period. Minimum is &quot;1 second&quot;.</td>
<td>Yes</td>
</tr>
<tr>
<td>@scheduledEndTime</td>
<td>The date and time that the run was scheduled to end.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@scheduledStartTime</td>
<td>The date and time that the run was scheduled to start.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@status</td>
<td>The status of this object. This is a runtime slot. Possible values are: pending, waiting_on_dependencies, running, waiting_on_runner, successful, and failed.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>
**MySqlDataNode**

Defines a data node using MySQL.

**Note**

The MySqlDataNode type is deprecated. While you can still use MySqlDataNode, it is recommended you use SqlDataNode. See the section called “SqlDataNode” (p. ?).

**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**

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>
## MySqlDataNode

This object includes the following fields from the `SqlDataNode` object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>connectionString</td>
<td>The JDBC connection string to access the database.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>insertQuery</td>
<td>A SQL statement to insert data into the table.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>*password</td>
<td>The password necessary to connect to the database.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>selectQuery</td>
<td>A SQL statement to fetch data from the table.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>table</td>
<td>The name of the table in the MySQL database.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>username</td>
<td>The user name necessary to connect to the database.</td>
<td>String</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This object includes the following fields from the `DataNode` object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>onFail</td>
<td>The SnsAlarm to use when the current instance fails.</td>
<td>SnsAlarm (p. 289) object reference</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>The SnsAlarm to use when the current instance succeeds.</td>
<td>SnsAlarm (p. 289) object reference</td>
<td>No</td>
</tr>
<tr>
<td>precondition</td>
<td>A list of precondition objects that must be true for the data node to be valid. A data node cannot reach the READY status until all its conditions are met. Preconditions do not have their own schedule or identity, instead they run on the schedule of the activity or data node with which they are associated.</td>
<td>A list of object references</td>
<td>No</td>
</tr>
<tr>
<td>schedule</td>
<td>A schedule of the object. A common use is to specify a time schedule that correlates to the schedule for the object. This slot overrides the schedule slot included from <code>SchedulableObject</code>, which is optional.</td>
<td>Schedule (p. 292) object reference</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### Schedule Type

Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of the interval or 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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>scheduleType</td>
<td>Required values are &quot;cron&quot; or &quot;timeseries&quot;. Defaults to &quot;timeseries&quot;.</td>
<td>Allowed values are &quot;cron&quot; or &quot;timeseries&quot;. Defaults to &quot;timeseries&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>workerGroup</td>
<td>The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields from RunnableObject.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>@activeInstances</td>
<td>Record of the currently scheduled instance objects</td>
<td>Schedulable object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptStatus</td>
<td>The status most recently reported from the object.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@actualEndTime</td>
<td>The date and time that the scheduled run actually ended. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@actualStartTime</td>
<td>The date and time that the scheduled run actually started. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptTimeout</td>
<td>The timeout time interval for an object attempt. If an attempt does not complete within the start time plus this time interval, AWS Data Pipeline marks the attempt as failed and your retry settings determine the next steps taken.</td>
<td>Time period; for example, &quot;1 hour&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@cascadeFailedOn</td>
<td>Description of which dependency the object failed on.</td>
<td>List of objects (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@componentParent</td>
<td>The component from which this instance is created.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorId</td>
<td>If the object failed, the error code. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorMessage</td>
<td>If the object failed, the error message. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorStackTrace</td>
<td>If the object failed, the error stack trace.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>failureAndRerunMode</td>
<td>Determines whether pipeline object failures and rerun commands cascade through pipeline object dependencies. For more information, see Cascading Failures and Reruns (p. 51).</td>
<td>String. Possible values are cascade and none.</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>@headAttempt</td>
<td>The latest attempt on the given instance.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@hostname</td>
<td>The host name of client that picked up the task attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>lateAfterTimeout</td>
<td>The time period in which the object run must start. If the object does not start within the scheduled start time plus this time interval, it is considered late.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>maximumRetries</td>
<td>The maximum number of times to retry the action. The default value is 2, which results in 3 tries total (1 original attempt plus 2 retries). The maximum value is 5 (6 total attempts).</td>
<td>Integer</td>
<td>No</td>
</tr>
<tr>
<td>onFail</td>
<td>An action to run when the current object fails.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onLateAction</td>
<td>The SnsAlarm to use when the object's run is late.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>An action to run when the current object succeeds.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>@reportProgressTime</td>
<td>The last time that Task Runner, or other code that is processing the tasks, called the ReportTaskProgress API.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>reportProgressTimeout</td>
<td>The time period for successive calls from Task Runner to the ReportTaskProgress API. If Task Runner, or other code that is processing the tasks, does not report progress within the specified period, the object attempt can be retried.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@resource</td>
<td>The resource instance on which the given activity/precondition attempt is being run.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>retryDelay</td>
<td>The timeout duration between two retry attempts. The default is 10 minutes.</td>
<td>Period. Minimum is &quot;1 second&quot;.</td>
<td>Yes</td>
</tr>
<tr>
<td>@scheduledEndTime</td>
<td>The date and time that the run was scheduled to end.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@scheduledStartTime</td>
<td>The date and time that the run was scheduled to start.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@status</td>
<td>The status of this object. This is a runtime slot. Possible values are: pending, waiting_on_dependencies, running, waiting_on_runner, successful, and failed.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>
## RedshiftDataNode

Defines a data node using Amazon Redshift.

### Example

The following is an example of this object type.

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

### Syntax

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### See Also

- [S3DataNode](p. 187)
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>createTableSql</td>
<td>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). Amazon EMR 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 is table 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.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>database</td>
<td>The database.</td>
<td>RedshiftDatabase (p.277) object reference</td>
<td>Yes</td>
</tr>
<tr>
<td>schemaName</td>
<td>This optional field specifies the name of the schema for the Amazon Redshift table. If not specified, Amazon EMR assumes that the schema name is PUBLIC, which is the default schema in Amazon Redshift. For more information, see Schemas in the Amazon Redshift Database Developer Guide.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>tableName</td>
<td>The name of the Amazon Redshift table. The table is created if it doesn't already exist and you've provided createTableSql.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>primaryKeys</td>
<td>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 Redshift table, this setting overrides the existing key.</td>
<td>List of Strings</td>
<td>No</td>
</tr>
</tbody>
</table>
This object includes the following fields from the `DataNode` object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>onFail</code></td>
<td>The SnsAlarm to use when the current instance fails.</td>
<td><code>SnsAlarm (p. 289)</code> object reference</td>
<td>No</td>
</tr>
<tr>
<td><code>onSuccess</code></td>
<td>The SnsAlarm to use when the current instance succeeds.</td>
<td><code>SnsAlarm (p. 289)</code> object reference</td>
<td>No</td>
</tr>
<tr>
<td><code>precondition</code></td>
<td>A list of precondition objects that must be true for the data node to be valid. A data node cannot reach the READY status until all its conditions are met. Preconditions do not have their own schedule or identity, instead they run on the schedule of the activity or data node with which they are associated.</td>
<td>A list of object references</td>
<td>No</td>
</tr>
<tr>
<td><code>schedule</code></td>
<td>A schedule of the object. A common use is to specify a time schedule that correlates to the schedule for the object. This slot overrides the schedule slot included from <code>SchedulableObject</code>, which is optional.</td>
<td><code>Schedule (p. 292)</code> object reference</td>
<td>Yes</td>
</tr>
<tr>
<td><code>scheduleType</code></td>
<td>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of the interval or 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.</td>
<td>Allowed values are &quot;cron&quot; or &quot;timeseries&quot;. Defaults to &quot;timeseries&quot;.</td>
<td>No</td>
</tr>
<tr>
<td><code>workerGroup</code></td>
<td>The worker group. This is used for routing tasks. If you provide a <code>runsOn</code> value and <code>workerGroup</code> exists, <code>workerGroup</code> is ignored.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields from `RunnableObject`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>@activeInstances</code></td>
<td>Record of the currently scheduled instance objects</td>
<td>Scheduleable object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td><code>attemptStatus</code></td>
<td>The status most recently reported from the object.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td><code>@actualEndTime</code></td>
<td>The date and time that the scheduled run actually ended. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td><code>@actualStartTime</code></td>
<td>The date and time that the scheduled run actually started. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>attemptTimeout</td>
<td>The timeout time interval for an object attempt. If an attempt does not complete within the start time plus this time interval, AWS Data Pipeline marks the attempt as failed and your retry settings determine the next steps taken.</td>
<td>Time period; for example, &quot;1 hour&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@cascadeFailedOn</td>
<td>Description of which dependency the object failed on.</td>
<td>List of objects (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@componentParent</td>
<td>The component from which this instance is created.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorId</td>
<td>If the object failed, the error code. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorMessage</td>
<td>If the object failed, the error message. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorStackTrace</td>
<td>If the object failed, the error stack trace.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>failureAndRerunMode</td>
<td>Determines whether pipeline object failures and rerun commands cascade through pipeline object dependencies. For more information, see Cascading Failures and Reruns (p. 51).</td>
<td>String. Possible values are cascade and none.</td>
<td>No</td>
</tr>
<tr>
<td>@headAttempt</td>
<td>The latest attempt on the given instance.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@hostname</td>
<td>The host name of client that picked up the task attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>lateAfterTimeout</td>
<td>The time period in which the object run must start. If the object does not start within the scheduled start time plus this time interval, it is considered late.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>maximumRetries</td>
<td>The maximum number of times to retry the action. The default value is 2, which results in 3 tries total (1 original attempt plus 2 retries). The maximum value is 5 (6 total attempts).</td>
<td>Integer</td>
<td>No</td>
</tr>
<tr>
<td>onFail</td>
<td>An action to run when the current object fails.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onLateAction</td>
<td>The SnsAlarm to use when the object's run is late.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>An action to run when the current object succeeds.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>@reportProgressTime</td>
<td>The last time that Task Runner, or other code that is processing the tasks, called the ReportTaskProgress API.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>reportProgressTimeout</td>
<td>The time period for successive calls from Task Runner to the ReportTaskProgress API. If Task Runner, or other code that is processing the tasks, does not report progress within the specified period, the object attempt can be retried.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@resource</td>
<td>The resource instance on which the given activity/precondition attempt is being run.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>retryDelay</td>
<td>The timeout duration between two retry attempts. The default is 10 minutes.</td>
<td>Period. Minimum is &quot;1 second&quot;.</td>
<td>Yes</td>
</tr>
<tr>
<td>@scheduledEndTime</td>
<td>The date and time that the run was scheduled to end.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@scheduledStartTime</td>
<td>The date and time that the run was scheduled to start.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@status</td>
<td>The status of this object. This is a runtime slot. Possible values are: pending, waiting_on_dependencies, running, waiting_on_runner, successful, and failed.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@triesLeft</td>
<td>The number of attempted runs remaining before setting the status of this object to failed. This is a runtime slot.</td>
<td>Integer (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@waitingOn</td>
<td>A list of all objects that this object is waiting on before it can enter the RUNNING state.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>

### S3DataNode

Defines a data node using Amazon S3.

**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

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>compression</td>
<td>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 when you use S3DataNode with CopyActivity.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>dataFormat</td>
<td>The format of the data described by the S3DataNode.</td>
<td>Data type object reference</td>
<td>Conditional</td>
</tr>
<tr>
<td>directoryPath</td>
<td>Amazon S3 directory path as a URI: s3://my-bucket/my-key-for-directory. You must provide either a filePath or directoryPath value.</td>
<td>String</td>
<td>Conditional</td>
</tr>
<tr>
<td>filePath</td>
<td>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 the directoryPath value to accommodate multiple files in a directory.</td>
<td>String</td>
<td>Conditional</td>
</tr>
</tbody>
</table>
### S3DataNode

This object includes the following fields from the `DataNode` object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>manifestFilePath</strong></td>
<td>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 <code>RedshiftCopyActivity</code> references the <code>S3DataNode</code>. For more information, see Using a manifest to specify data files.</td>
<td>String</td>
<td>Conditional</td>
</tr>
<tr>
<td><strong>onFail</strong></td>
<td>The SnsAlarm to use when the current instance fails.</td>
<td>SnsAlarm (p. 289) object reference</td>
<td>No</td>
</tr>
<tr>
<td><strong>onSuccess</strong></td>
<td>The SnsAlarm to use when the current instance succeeds.</td>
<td>SnsAlarm (p. 289) object reference</td>
<td>No</td>
</tr>
<tr>
<td><strong>precondition</strong></td>
<td>A list of precondition objects that must be true for the data node to be valid. A data node cannot reach the READY status until all its conditions are met. Preconditions do not have their own schedule or identity, instead they run on the schedule of the activity or data node with which they are associated.</td>
<td>A list of object references</td>
<td>No</td>
</tr>
<tr>
<td><strong>schedule</strong></td>
<td>A schedule of the object. A common use is to specify a time schedule that correlates to the schedule for the object. This slot overrides the schedule slot included from <code>SchedulableObject</code>, which is optional.</td>
<td>Schedule (p. 292) object reference</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>scheduleType</strong></td>
<td>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of the interval or 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.</td>
<td>Allowed values are &quot;cron&quot; or &quot;timeseries&quot;. Defaults to &quot;timeseries&quot;</td>
<td>No</td>
</tr>
<tr>
<td><strong>workerGroup</strong></td>
<td>The worker group. This is used for routing tasks. If you provide a <code>runsOn</code> value and <code>workerGroup exists</code>, <code>workerGroup</code> is ignored.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>
This object includes the following fields from `RunnableObject`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>@activeInstances</td>
<td>Record of the currently scheduled instance objects</td>
<td>Schedulable object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptStatus</td>
<td>The status most recently reported from the object.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@actualEndTime</td>
<td>The date and time that the scheduled run actually ended. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@actualStartTime</td>
<td>The date and time that the scheduled run actually started. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptTimeout</td>
<td>The timeout time interval for an object attempt. If an attempt does not complete within the start time plus this time interval, AWS Data Pipeline marks the attempt as failed and your retry settings determine the next steps taken.</td>
<td>Time period; for example, “1 hour”.</td>
<td>No</td>
</tr>
<tr>
<td>@cascadeFailedOn</td>
<td>Description of which dependency the object failed on.</td>
<td>List of objects (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@componentParent</td>
<td>The component from which this instance is created.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorId</td>
<td>If the object failed, the error code. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorMessage</td>
<td>If the object failed, the error message. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorStackTrace</td>
<td>If the object failed, the error stack trace.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>failureAndRerunMode</td>
<td>Determines whether pipeline object failures and rerun commands cascade through pipeline object dependencies. For more information, see Cascading Failures and Reruns (p. 51).</td>
<td>String. Possible values are <code>cascade</code> and <code>none</code>.</td>
<td>No</td>
</tr>
<tr>
<td>@headAttempt</td>
<td>The latest attempt on the given instance.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@hostname</td>
<td>The host name of client that picked up the task attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>lateAfterTimeout</td>
<td>The time period in which the object run must start. If the object does not start within the scheduled start time plus this time interval, it is considered late.</td>
<td>Time period; for example, “1 hour”. The minimum value is “15 minutes”.</td>
<td>No</td>
</tr>
<tr>
<td>maximumRetries</td>
<td>The maximum number of times to retry the action. The default value is 2, which results in 3 tries total (1 original attempt plus 2 retries). The maximum value is 5 (6 total attempts).</td>
<td>Integer</td>
<td>No</td>
</tr>
</tbody>
</table>
# SqlDataNode

Defines a data node using SQL.

## Name | Description | Type | Required
--- | --- | --- | ---
oneFail | An action to run when the current object fails. | List of SnsAlarm (p. 289) object references | No
oneLateAction | The SnsAlarm to use when the object's run is late. | List of SnsAlarm (p. 289) object references | No
onSuccess | An action to run when the current object succeeds. | List of SnsAlarm (p. 289) object references | No
@reportProgressTime | The last time that Task Runner, or other code that is processing the tasks, called the ReportTaskProgress API. | DateTime (read-only) | No
reportProgressTimeout | The time period for successive calls from Task Runner to the ReportTaskProgress API. If Task Runner, or other code that is processing the tasks, does not report progress within the specified period, the object attempt can be retried. | Time period; for example, “1 hour”. The minimum value is "15 minutes". | No
@resource | The resource instance on which the given activity/precondition attempt is being run. | Object reference (read-only) | No
retryDelay | The timeout duration between two retry attempts. The default is 10 minutes. | Period. Minimum is "1 second". | Yes
@scheduledEndTime | The date and time that the run was scheduled to end. | DateTime (read-only) | No
@scheduledStartTime | The date and time that the run was scheduled to start. | DateTime (read-only) | No
@status | The status of this object. This is a runtime slot. Possible values are: pending, waiting_on_dependencies, running, waiting_on_runner, successful, and failed. | String (read-only) | No
@triesLeft | The number of attempted runs remaining before setting the status of this object to failed. This is a runtime slot. | Integer (read-only) | No
@waitingOn | A list of all objects that this object is waiting on before it can enter the RUNNING state. | String (read-only) | No

## See Also
- MySqlDataNode (p. 179)
Note
The SqlDataNode type only supports MySQL.

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.

```json
{
  "id" : "Sql Table",
  "type" : "SqlDataNode",
  "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
The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>
This object includes the following fields from the `SqlDataNode` object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>connectionString</td>
<td>The JDBC connection string to access the database.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>insertQuery</td>
<td>A SQL statement to insert data into the table.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>*password</td>
<td>The password necessary to connect to the database.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>selectQuery</td>
<td>A SQL statement to fetch data from the table.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>table</td>
<td>The name of the table in the MySQL database.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>username</td>
<td>The user name necessary to connect to the database.</td>
<td>String</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This object includes the following fields from the `DataNode` object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>onFail</td>
<td>The SnsAlarm to use when the current instance fails.</td>
<td>SnsAlarm (p. 289) object reference</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>The SnsAlarm to use when the current instance succeeds.</td>
<td>SnsAlarm (p. 289) object reference</td>
<td>No</td>
</tr>
<tr>
<td>precondition</td>
<td>A list of precondition objects that must be true for the data node to be valid. A data node cannot reach the READY status until all its conditions are met. Preconditions do not have their own schedule or identity, instead they run on the schedule of the activity or data node with which they are associated.</td>
<td>A list of object references</td>
<td>No</td>
</tr>
<tr>
<td>schedule</td>
<td>A schedule of the object. A common use is to specify a time schedule that correlates to the schedule for the object. This slot overrides the schedule slot included from SchedulableObject, which is optional.</td>
<td>Schedule (p. 292) object reference</td>
<td>Yes</td>
</tr>
<tr>
<td>scheduleType</td>
<td>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of the interval or 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.</td>
<td>Allowed values are &quot;cron&quot; or &quot;timeseries&quot;. Defaults to &quot;timeseries&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>workerGroup</td>
<td>The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields from `RunnableObject`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>@activeInstances</td>
<td>Record of the currently scheduled instance objects</td>
<td>Schedulable object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptStatus</td>
<td>The status most recently reported from the object.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@actualEndTime</td>
<td>The date and time that the scheduled run actually ended. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@actualStartTime</td>
<td>The date and time that the scheduled run actually started. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptTimeout</td>
<td>The timeout time interval for an object attempt. If an attempt does not complete within the start time plus this time interval, AWS Data Pipeline marks the attempt as failed and your retry settings determine the next steps taken.</td>
<td>Time period; for example, &quot;1 hour&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@cascadeFailedOn</td>
<td>Description of which dependency the object failed on.</td>
<td>List of objects (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@componentParent</td>
<td>The component from which this instance is created.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorId</td>
<td>If the object failed, the error code. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorMessage</td>
<td>If the object failed, the error message. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorStackTrace</td>
<td>If the object failed, the error stack trace.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>failureAndRerunMode</td>
<td>Determines whether pipeline object failures and rerun commands cascade through pipeline object dependencies. For more information, see [Cascading Failures and Reruns](p. 51).</td>
<td>String. Possible values are <code>cascade</code> and <code>none</code>.</td>
<td>No</td>
</tr>
<tr>
<td>@headAttempt</td>
<td>The latest attempt on the given instance.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@hostname</td>
<td>The host name of client that picked up the task attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>lateAfterTimeout</td>
<td>The time period in which the object run must start. If the object does not start within the scheduled start time plus this time interval, it is considered late.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>maximumRetries</td>
<td>The maximum number of times to retry the action. The default value is 2, which results in 3 tries total (1 original attempt plus 2 retries). The maximum value is 5 (6 total attempts).</td>
<td>Integer</td>
<td>No</td>
</tr>
<tr>
<td>onFail</td>
<td>An action to run when the current object fails.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onLateAction</td>
<td>The SnsAlarm to use when the object's run is late.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>An action to run when the current object succeeds.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>@reportProgressTime</td>
<td>The last time that Task Runner, or other code that is processing the tasks, called the ReportTaskProgress API.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>reportProgressTimeout</td>
<td>The time period for successive calls from Task Runner to the ReportTaskProgress API. If Task Runner, or other code that is processing the tasks, does not report progress within the specified period, the object attempt can be retried.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@resource</td>
<td>The resource instance on which the given activity/precondition attempt is being run.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>retryDelay</td>
<td>The timeout duration between two retry attempts. The default is 10 minutes.</td>
<td>Period. Minimum is &quot;1 second&quot;.</td>
<td>Yes</td>
</tr>
<tr>
<td>@scheduledEndTime</td>
<td>The date and time that the run was scheduled to end.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@scheduledStartTime</td>
<td>The date and time that the run was scheduled to start.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@status</td>
<td>The status of this object. This is a runtime slot. Possible values are: pending, waiting_on_dependencies, running, waiting_on_runner, successful, and failed.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@triesLeft</td>
<td>The number of attempted runs remaining before setting the status of this object to failed. This is a runtime slot.</td>
<td>Integer (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@waitingOn</td>
<td>A list of all objects that this object is waiting on before it can enter the RUNNING state.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>
See Also

- S3DataNode (p. 187)

Activities

The following are Data Pipeline Activities:

Topics

- CopyActivity (p. 196)
- EmrActivity (p. 201)
- HiveActivity (p. 207)
- HiveCopyActivity (p. 212)
- PigActivity (p. 218)
- RedshiftCopyActivity (p. 227)
- ShellCommandActivity (p. 233)
- SqlActivity (p. 239)

CopyActivity

Copies data from one location to another. CopyActivity supports S3DataNode (p. 187) and MySqlDataNode (p. 179) 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
- Each file is smaller than 4 GB

Attempting to copy files larger than 4 GB causes an error during task execution. Additionally, you may encounter repeated CopyActivity failures if you supply compressed data files as input and do not specify this using the compression field on the S3 data nodes. 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 specifies that:

- 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").

Important

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.

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" }
}
```

Syntax

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>The input data source.</td>
<td>Data node object reference</td>
<td>Yes</td>
</tr>
<tr>
<td>output</td>
<td>The location for the output.</td>
<td>Data node object reference</td>
<td>Yes</td>
</tr>
</tbody>
</table>
This object includes the following fields from the Activity object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>dependsOn</td>
<td>One or more references to other Activities that must reach the FINISHED state before this activity will start.</td>
<td>Activity object reference</td>
<td>No</td>
</tr>
<tr>
<td>onFail</td>
<td>The SnsAlarm to use when the current instance fails.</td>
<td>SnsAlarm (p. 289) object reference</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>The SnsAlarm to use when the current instance succeeds.</td>
<td>SnsAlarm (p. 289) object reference</td>
<td>No</td>
</tr>
<tr>
<td>precondition</td>
<td>A condition that must be met before the object can run. To specify multiple conditions, add multiple precondition fields. The activity cannot run until all its conditions are met.</td>
<td>List of preconditions</td>
<td>No</td>
</tr>
<tr>
<td>schedule</td>
<td>A schedule of the object. A common use is to specify a time schedule that correlates to the schedule for the object. This slot overrides the schedule slot included from SchedulableObject, which is optional.</td>
<td>Schedule (p. 292) object reference</td>
<td>Yes</td>
</tr>
<tr>
<td>scheduleType</td>
<td>Specifies whether the pipeline component should be scheduled at the beginning of the interval or the end of the interval. timeseries means instances are scheduled at the end of each interval and cron means instances are scheduled at the beginning of each interval. The default value is timeseries.</td>
<td>Schedule (p. 292) object reference</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields from RunnableObject.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>@activeInstances</td>
<td>Record of the currently scheduled instance objects</td>
<td>Schedulable object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptStatus</td>
<td>The status most recently reported from the object.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@actualEndTime</td>
<td>The date and time that the scheduled run actually ended. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@actualStartTime</td>
<td>The date and time that the scheduled run actually started. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>attemptTimeout</td>
<td>The timeout time interval for an object attempt. If an attempt does not complete within the start time plus this time interval, AWS Data Pipeline marks the attempt as failed and your retry settings determine the next steps taken.</td>
<td>Time period; for example, &quot;1 hour&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@cascadeFailedOn</td>
<td>Description of which dependency the object failed on.</td>
<td>List of objects (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@componentParent</td>
<td>The component from which this instance is created.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorId</td>
<td>If the object failed, the error code. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorMessage</td>
<td>If the object failed, the error message. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorStackTrace</td>
<td>If the object failed, the error stack trace.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>failureAndRerunMode</td>
<td>Determines whether pipeline object failures and rerun commands cascade through pipeline object dependencies. For more information, see Cascading Failures and Reruns (p. 51).</td>
<td>String. Possible values are cascade and none.</td>
<td>No</td>
</tr>
<tr>
<td>@headAttempt</td>
<td>The latest attempt on the given instance.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@hostname</td>
<td>The host name of client that picked up the task attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>lateAfterTimeout</td>
<td>The time period in which the object run must start. If the object does not start within the scheduled start time plus this time interval, it is considered late.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;</td>
<td>No</td>
</tr>
<tr>
<td>maximumRetries</td>
<td>The maximum number of times to retry the action. The default value is 2, which results in 3 tries total (1 original attempt plus 2 retries). The maximum value is 5 (6 total attempts).</td>
<td>Integer</td>
<td>No</td>
</tr>
<tr>
<td>onFail</td>
<td>An action to run when the current object fails.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onLateAction</td>
<td>The SnsAlarm to use when the object's run is late.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>An action to run when the current object succeeds.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>@reportProgressTime</td>
<td>The last time that Task Runner, or other code that is processing the tasks, called the ReportTaskProgress API.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>reportProgressTimeout</td>
<td>The time period for successive calls from Task Runner to the ReportTaskProgress API. If Task Runner, or other code that is processing the tasks, does not report progress within the specified period, the object attempt can be retried.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@resource</td>
<td>The resource instance on which the given activity/precondition attempt is being run.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>retryDelay</td>
<td>The timeout duration between two retry attempts. The default is 10 minutes.</td>
<td>Period. Minimum is &quot;1 second&quot;.</td>
<td>Yes</td>
</tr>
<tr>
<td>@scheduledEndTime</td>
<td>The date and time that the run was scheduled to end.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@scheduledStartTime</td>
<td>The date and time that the run was scheduled to start.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@status</td>
<td>The status of this object. This is a runtime slot. Possible values are: pending, waiting_on_dependencies, running, waiting_on_runner, successful, and failed.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@triesLeft</td>
<td>The number of attempted runs remaining before setting the status of this object to failed. This is a runtime slot.</td>
<td>Integer (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@waitingOn</td>
<td>A list of all objects that this object is waiting on before it can enter the RUNNING state.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields from SchedulableObject.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxActiveInstances</td>
<td>The maximum number of concurrent active instances of a component. For activities, setting this to 1 runs instances in strict chronological order. A value greater than 1 allows different instances of the activity to run concurrently and requires you to ensure your activity can tolerate concurrent execution.</td>
<td>Integer between 1 and 5</td>
<td>No</td>
</tr>
<tr>
<td>runsOn</td>
<td>The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.</td>
<td>Resource object reference</td>
<td>No</td>
</tr>
<tr>
<td>schedule</td>
<td>A schedule of the object. A common use is to specify a time schedule that correlates to the schedule for the object.</td>
<td>Schedule (p. 292) object reference</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>scheduleType</td>
<td>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of the interval or 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.</td>
<td>Allowed values are &quot;cron&quot; or &quot;timeseries&quot;. Defaults to &quot;timeseries&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>workerGroup</td>
<td>The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>healthStatus</td>
<td>The health status of the object, which reflects success or failure of the last instance that reached a terminated state. Values are: HEALTHY or ERROR.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>healthStatusFromInstanceId</td>
<td>The ID of the last object instance that reached a terminated state.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>healthStatusUpdatedTime</td>
<td>The last time at which the health status was updated.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>

**See Also**

- ShellCommandActivity (p. 233)
- EmrActivity (p. 201)
- Export MySQL Data to Amazon S3 with CopyActivity (p. 103)
- Copy Data from Amazon S3 to MySQL (p. 316)

**EmrActivity**

Runs an Amazon 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 an Amazon EMR-formatted step, followed by its AWS Data Pipeline equivalent.

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

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

**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. 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 Pig script, Hadoop streaming cluster, your own custom JAR including its parameters, etc.

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

Syntax
The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>actionOnResourceFailure</td>
<td>Action for the EmrCluster to take when it fails.</td>
<td>String: retryall (retry all inputs) or retyrnone (retry nothing)</td>
<td>No</td>
</tr>
<tr>
<td>actionOnTaskFailure</td>
<td>Action for the activity/task to take when its associated EmrCluster fails.</td>
<td>String: continue (do not terminate the cluster) or terminate</td>
<td>No</td>
</tr>
</tbody>
</table>
## EmrActivity

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>The input data source.</td>
<td>Data node object reference</td>
<td>No</td>
</tr>
<tr>
<td>output</td>
<td>The location for the output.</td>
<td>Data node object reference</td>
<td>No</td>
</tr>
<tr>
<td>preStepCommand</td>
<td>Shell scripts to be run before any steps are run.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>postStepCommand</td>
<td>Shell scripts to be run after all steps are finished.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>runsOn</td>
<td>The Amazon EMR cluster to run this cluster.</td>
<td>EmrCluster (p. 250) object reference</td>
<td>Yes</td>
</tr>
<tr>
<td>step</td>
<td>One or more steps for the cluster to run.</td>
<td>String</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This object includes the following fields from the Activity object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>dependsOn</td>
<td>One or more references to other Activities that must reach the FINISHED state before this activity will start.</td>
<td>Activity object reference</td>
<td>No</td>
</tr>
<tr>
<td>onFail</td>
<td>The SnsAlarm to use when the current instance fails.</td>
<td>SnsAlarm (p. 289) object reference</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>The SnsAlarm to use when the current instance succeeds.</td>
<td>SnsAlarm (p. 289) object reference</td>
<td>No</td>
</tr>
<tr>
<td>precondition</td>
<td>A condition that must be met before the object can run. To specify multiple conditions, add multiple precondition fields. The activity cannot run until all its conditions are met.</td>
<td>List of preconditions</td>
<td>No</td>
</tr>
<tr>
<td>schedule</td>
<td>A schedule of the object. A common use is to specify a time schedule that correlates to the schedule for the object. This slot overrides the schedule slot included from SchedulableObject, which is optional.</td>
<td>Schedule (p. 292) object reference</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### scheduleType

Specifies whether the pipeline component should be scheduled at the beginning of the interval or the end of the interval. **timeseries** means instances are scheduled at the end of each interval and **cron** means instances are scheduled at the beginning of each interval. The default value is **timeseries**.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>scheduleType</td>
<td>Specifies whether the pipeline component should be scheduled at the beginning of the interval or the end of the interval. <strong>Timeseries</strong> means instances are scheduled at the end of each interval and <strong>cron</strong> means instances are scheduled at the beginning of each interval. The default value is <strong>timeseries</strong>.</td>
<td>Schedule (p. 292) object reference</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields from **RunnableObject**.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>@activeInstances</td>
<td>Record of the currently scheduled instance objects</td>
<td>Scheduled object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptStatus</td>
<td>The status most recently reported from the object.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@actualEndTime</td>
<td>The date and time that the scheduled run actually ended. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@actualStartTime</td>
<td>The date and time that the scheduled run actually started. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptTimeout</td>
<td>The timeout time interval for an object attempt. If an attempt does not complete within the start time plus this time interval, AWS Data Pipeline marks the attempt as failed and your retry settings determine the next steps taken.</td>
<td>Time period; for example, &quot;1 hour&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@cascadeFailedOn</td>
<td>Description of which dependency the object failed on.</td>
<td>List of objects (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@componentParent</td>
<td>The component from which this instance is created.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorId</td>
<td>If the object failed, the error code. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorMessage</td>
<td>If the object failed, the error message. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorStackTrace</td>
<td>If the object failed, the error stack trace.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>failureAndRerunMode</td>
<td>Determines whether pipeline object failures and rerun commands cascade through pipeline object dependencies. For more information, see Cascading Failures and Reruns (p. 51).</td>
<td>String. Possible values are <strong>cascade</strong> and <strong>none</strong>.</td>
<td>No</td>
</tr>
<tr>
<td>@headAttempt</td>
<td>The latest attempt on the given instance.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@hostname</td>
<td>The host name of client that picked up the task attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>lateAfterTimeout</td>
<td>The time period in which the object run must start. If the object does not start within the scheduled start time plus this time interval, it is considered late.</td>
<td>Time period; for example, &quot;1 hour&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>maximumRetries</td>
<td>The maximum number of times to retry the action. The default value is 2, which results in 3 tries total (1 original attempt plus 2 retries). The maximum value is 5 (6 total attempts).</td>
<td>Integer</td>
<td>No</td>
</tr>
<tr>
<td>onFail</td>
<td>An action to run when the current object fails.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onLateAction</td>
<td>The SnsAlarm to use when the object's run is late.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>An action to run when the current object succeeds.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>@reportProgressTime</td>
<td>The last time that Task Runner, or other code that is processing the tasks, called the ReportTaskProgress API.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>reportProgressTimeout</td>
<td>The time period for successive calls from Task Runner to the ReportTaskProgress API. If Task Runner, or other code that is processing the tasks, does not report progress within the specified period, the object attempt can be retried.</td>
<td>Time period; for example, &quot;1 hour&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@resource</td>
<td>The resource instance on which the given activity/precondition attempt is being run.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>retryDelay</td>
<td>The timeout duration between two retry attempts. The default is 10 minutes.</td>
<td>Period. Minimum is &quot;1 second&quot;</td>
<td>Yes</td>
</tr>
<tr>
<td>@scheduledEndTime</td>
<td>The date and time that the run was scheduled to end.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@scheduledStartTime</td>
<td>The date and time that the run was scheduled to start.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@status</td>
<td>The status of this object. This is a runtime slot. Possible values are: pending, waiting_on_dependencies, running, waiting_on_runner, successful, and failed.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@triesLeft</td>
<td>The number of attempted runs remaining before setting the status of this object to failed. This is a runtime slot.</td>
<td>Integer (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@waitingOn</td>
<td>A list of all objects that this object is waiting on before it can enter the RUNNING state.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>
This object includes the following fields from `SchedulableObject`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxActiveInstances</td>
<td>The maximum number of concurrent active instances of a component. For activities, setting this to 1 runs instances in strict chronological order. A value greater than 1 allows different instances of the activity to run concurrently and requires you to ensure your activity can tolerate concurrent execution.</td>
<td>Integer between 1 and 5</td>
<td>No</td>
</tr>
<tr>
<td>runsOn</td>
<td>The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.</td>
<td>Resource object reference</td>
<td>No</td>
</tr>
<tr>
<td>schedule</td>
<td>A schedule of the object. A common use is to specify a time schedule that correlates to the schedule for the object.</td>
<td>Schedule (p. 292) object reference</td>
<td>No</td>
</tr>
<tr>
<td>scheduleType</td>
<td>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of the interval or 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.</td>
<td>Allowed values are &quot;cron&quot; or &quot;timeseries&quot;. Defaults to &quot;timeseries&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>workerGroup</td>
<td>The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>healthStatus</td>
<td>The health status of the object, which reflects success or failure of the last instance that reached a terminated state. Values are: HEALTHY or ERROR.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>healthStatusFromInstanceId</td>
<td>The ID of the last object instance that reached a terminated state.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>healthStatusUpdatedTime</td>
<td>The last time at which the health status was updated.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>

See Also

- `ShellCommandActivity (p. 233)`
- `CopyActivity (p. 196)`
- `EmrCluster (p. 250)`
HiveActivity

Runs a Hive query on an Amazon 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}`, etc. based on the input fields in the HiveActivity object. For S3 inputs, the dataFormat field is used to create the Hive column names. For MySQL (RDS) inputs, the column names for the SQL query are used to create the Hive column names.

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.

```json
{
    "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

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>
This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>generatedScriptsPath</td>
<td>An Amazon S3 path capturing the Hive script that ran after all the expressions in it were evaluated, including staging information. This script is stored for troubleshooting purposes.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>hiveScript</td>
<td>The Hive script to run.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>input</td>
<td>The input data source.</td>
<td>Data node object reference</td>
<td>Yes</td>
</tr>
<tr>
<td>output</td>
<td>The location for the output.</td>
<td>Data node object reference</td>
<td>Yes</td>
</tr>
<tr>
<td>runsOn</td>
<td>The Amazon EMR cluster to run this activity.</td>
<td>EmrCluster (p. 250) object reference</td>
<td>Yes</td>
</tr>
<tr>
<td>scriptUri</td>
<td>The location of the Hive script to run. For example, s3://script location.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>
| 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 stage set to true or false.

This field is especially useful to send dynamic values to Hive using AWS Data Pipeline expressions and functions. For more information, see Pipeline Expressions and Functions (p. 161). | String                        | No       |
| stage              | Determines whether staging is enabled.                                                                                                                                                                     | Boolean                       | No       |

**Note**

You must specify a hiveScript value or a scriptUri value, but both are not required.

This object includes the following fields from the Activity object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>dependsOn</td>
<td>One or more references to other Activities that must reach the FINISHED state before this activity will start.</td>
<td>Activity object reference</td>
<td>No</td>
</tr>
<tr>
<td>onFail</td>
<td>The SnsAlarm to use when the current instance fails.</td>
<td>SnsAlarm (p. 289) object reference</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>onSuccess</td>
<td>The SnsAlarm to use when the current instance succeeds.</td>
<td>SnsAlarm (p. 289) object reference</td>
<td>No</td>
</tr>
<tr>
<td>precondition</td>
<td>A condition that must be met before the object can run. To specify multiple conditions, add multiple precondition fields. The activity cannot run until all its conditions are met.</td>
<td>List of preconditions</td>
<td>No</td>
</tr>
<tr>
<td>schedule</td>
<td>A schedule of the object. A common use is to specify a time schedule that correlates to the schedule for the object. This slot overrides the schedule slot included from SchedulableObject, which is optional.</td>
<td>Schedule (p. 292) object reference</td>
<td>Yes</td>
</tr>
<tr>
<td>scheduleType</td>
<td>Specifies whether the pipeline component should be scheduled at the beginning of the interval or the end of the interval. timeseries means instances are scheduled at the end of each interval and cron means instances are scheduled at the beginning of each interval. The default value is timeseries.</td>
<td>Schedule (p. 292) object reference</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields from RunnableObject.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>@activeInstances</td>
<td>Record of the currently scheduled instance objects</td>
<td>Schedulable object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptStatus</td>
<td>The status most recently reported from the object.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@actualEndTime</td>
<td>The date and time that the scheduled run actually ended. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@actualStartTime</td>
<td>The date and time that the scheduled run actually started. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptTimeout</td>
<td>The timeout time interval for an object attempt. If an attempt does not complete within the start time plus this time interval, AWS Data Pipeline marks the attempt as failed and your retry settings determine the next steps taken.</td>
<td>Time period; for example, “1 hour”.</td>
<td>No</td>
</tr>
<tr>
<td>@cascadeFailedOn</td>
<td>Description of which dependency the object failed on.</td>
<td>List of objects (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@componentParent</td>
<td>The component from which this instance is created.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorId</td>
<td>If the object failed, the error code. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>errorMessage</td>
<td>If the object failed, the error message. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorStackTrace</td>
<td>If the object failed, the error stack trace.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>failureAndRerunMode</td>
<td>Determines whether pipeline object failures and rerun commands cascade through pipeline object dependencies. For more information, see Cascading Failures and Reruns (p. 51).</td>
<td>String. Possible values are <code>cascade</code> and <code>none</code>.</td>
<td>No</td>
</tr>
<tr>
<td>@headAttempt</td>
<td>The latest attempt on the given instance.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@hostname</td>
<td>The host name of client that picked up the task attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>lateAfterTimeout</td>
<td>The time period in which the object run must start. If the object does not start within the scheduled start time plus this time interval, it is considered late.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>maximumRetries</td>
<td>The maximum number of times to retry the action. The default value is 2, which results in 3 tries total (1 original attempt plus 2 retries). The maximum value is 5 (6 total attempts).</td>
<td>Integer</td>
<td>No</td>
</tr>
<tr>
<td>onFail</td>
<td>An action to run when the current object fails.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onLateAction</td>
<td>The SnsAlarm to use when the object’s run is late.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>An action to run when the current object succeeds.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>@reportProgressTime</td>
<td>The last time that Task Runner, or other code that is processing the tasks, called the ReportTaskProgress API.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>reportProgressTimeout</td>
<td>The time period for successive calls from Task Runner to the ReportTaskProgress API. If Task Runner, or other code that is processing the tasks, does not report progress within the specified period, the object attempt can be retried.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@resource</td>
<td>The resource instance on which the given activity/precondition attempt is being run.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>retryDelay</td>
<td>The timeout duration between two retry attempts. The default is 10 minutes.</td>
<td>Period. Minimum is &quot;1 second&quot;.</td>
<td>Yes</td>
</tr>
<tr>
<td>@scheduledEndTime</td>
<td>The date and time that the run was scheduled to end.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>
This object includes the following fields from `SortableObject`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>maxActiveInstances</code></td>
<td>The maximum number of concurrent active instances of a component. For activities, setting this to 1 runs instances in strict chronological order. A value greater than 1 allows different instances of the activity to run concurrently and requires you to ensure your activity can tolerate concurrent execution.</td>
<td>Integer between 1 and 5</td>
<td>No</td>
</tr>
<tr>
<td><code>runsOn</code></td>
<td>The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.</td>
<td>Resource object reference</td>
<td>No</td>
</tr>
<tr>
<td><code>schedule</code></td>
<td>A schedule of the object. A common use is to specify a time schedule that correlates to the schedule for the object.</td>
<td><code>Schedule (p. 292)</code> object reference</td>
<td>No</td>
</tr>
<tr>
<td><code>scheduleType</code></td>
<td>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of the interval or 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.</td>
<td>Allowed values are &quot;cron&quot; or &quot;timeseries&quot;. Defaults to &quot;timeseries&quot;.</td>
<td>No</td>
</tr>
<tr>
<td><code>workerGroup</code></td>
<td>The worker group. This is used for routing tasks. If you provide a <code>runsOn</code> value and <code>workerGroup</code> exists, <code>workerGroup</code> is ignored.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td><code>healthStatus</code></td>
<td>The health status of the object, which reflects success or failure of the last instance that reached a terminated state. Values are: <code>HEALTHY</code> or <code>ERROR</code>.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>
HiveCopyActivity

Runs a Hive query on an Amazon EMR cluster. HiveCopyActivity makes it easier to copy data between Amazon S3 and DynamoDB. HiveCopyActivity accepts a HiveQL statement to filter input data from Amazon S3 or 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.

```json
{
  "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" }
    }
  ]
}
```
Syntax

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>
This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>filterSql</td>
<td>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.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>generatedScriptsPath</td>
<td>An Amazon S3 path capturing the Hive script that ran after all the expressions in it were evaluated, including staging information. This script is stored for troubleshooting purposes.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>input</td>
<td>The input data node. This must be S3DataNode (p. 187) or DynamoDBDataNode (p. 174). If you use DynamoDBDataNode, specify a DynamoDBExportDataFormat (p. 284).</td>
<td>S3DataNode (p. 187) or DynamoDBDataNode (p. 174)</td>
<td>Yes</td>
</tr>
<tr>
<td>output</td>
<td>The output data node. If input is S3DataNode (p. 187), this must be DynamoDBDataNode (p. 174). Otherwise, this can be S3DataNode (p. 187) or DynamoDBDataNode (p. 174). If you use DynamoDBDataNode, specify a DynamoDBExportDataFormat (p. 284).</td>
<td>S3DataNode (p. 187) or DynamoDBDataNode (p. 174)</td>
<td>Yes</td>
</tr>
<tr>
<td>runsOn</td>
<td>The Amazon EMR cluster to run this activity.</td>
<td>EmrCluster (p. 250)</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This object includes the following fields from the Activity object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>dependsOn</td>
<td>One or more references to other Activities that must reach the FINISHED state before this activity will start.</td>
<td>Activity object reference</td>
<td>No</td>
</tr>
<tr>
<td>onFail</td>
<td>The SnsAlarm to use when the current instance fails.</td>
<td>SnsAlarm (p. 289) object reference</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>The SnsAlarm to use when the current instance succeeds.</td>
<td>SnsAlarm (p. 289) object reference</td>
<td>No</td>
</tr>
<tr>
<td>precondition</td>
<td>A condition that must be met before the object can run. To specify multiple conditions, add multiple precondition fields. The activity cannot run until all its conditions are met.</td>
<td>List of preconditions</td>
<td>No</td>
</tr>
</tbody>
</table>
### Schedule

**Name**: schedule

A schedule of the object. A common use is to specify a time schedule that correlates to the schedule for the object.

This slot overrides the schedule slot included from SchedulableObject, which is optional.

**Type**: Schedule (p. 292) object reference

**Required**: Yes

### ScheduleType

**Name**: scheduleType

Specifies whether the pipeline component should be scheduled at the beginning of the interval or the end of the interval.

*timeseries* means instances are scheduled at the end of each interval and *cron* means instances are scheduled at the beginning of each interval. The default value is *timeseries*.

**Type**: Schedule (p. 292) object reference

**Required**: No

This object includes the following fields from RunnableObject.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>@activeInstances</td>
<td>Record of the currently scheduled instance objects</td>
<td>Schedulable object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptStatus</td>
<td>The status most recently reported from the object.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@actualEndTime</td>
<td>The date and time that the scheduled run actually ended. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@actualStartTime</td>
<td>The date and time that the scheduled run actually started. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptTimeout</td>
<td>The timeout time interval for an object attempt. If an attempt does not complete within the start time plus this time interval, AWS Data Pipeline marks the attempt as failed and your retry settings determine the next steps taken.</td>
<td>Time period; for example, &quot;1 hour&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@cascadeFailedOn</td>
<td>Description of which dependency the object failed on.</td>
<td>List of objects (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@componentParent</td>
<td>The component from which this instance is created.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorId</td>
<td>If the object failed, the error code. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorMessage</td>
<td>If the object failed, the error message. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorStackTrace</td>
<td>If the object failed, the error stack trace.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>failureAndRerunMode</td>
<td>Determines whether pipeline object failures and rerun commands cascade through pipeline object dependencies. For more information, see Cascading Failures and Reruns (p. 51).</td>
<td>String. Possible values are cascade and none.</td>
<td>No</td>
</tr>
<tr>
<td>@headAttempt</td>
<td>The latest attempt on the given instance.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@hostname</td>
<td>The host name of client that picked up the task attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>lateAfterTimeout</td>
<td>The time period in which the object run must start. If the object does not start within the scheduled start time plus this time interval, it is considered late.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>maximumRetries</td>
<td>The maximum number of times to retry the action. The default value is 2, which results in 3 tries total (1 original attempt plus 2 retries). The maximum value is 5 (6 total attempts).</td>
<td>Integer</td>
<td>No</td>
</tr>
<tr>
<td>onFail</td>
<td>An action to run when the current object fails.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onLateAction</td>
<td>The SnsAlarm to use when the object’s run is late.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>An action to run when the current object succeeds.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>@reportProgressTime</td>
<td>The last time that Task Runner, or other code that is processing the tasks, called the ReportTaskProgress API.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>reportProgressTimeout</td>
<td>The time period for successive calls from Task Runner to the ReportTaskProgress API. If Task Runner, or other code that is processing the tasks, does not report progress within the specified period, the object attempt can be retried.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@resource</td>
<td>The resource instance on which the given activity/precondition attempt is being run.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>retryDelay</td>
<td>The timeout duration between two retry attempts. The default is 10 minutes.</td>
<td>Period. Minimum is &quot;1 second&quot;.</td>
<td>Yes</td>
</tr>
<tr>
<td>@scheduledEndTime</td>
<td>The date and time that the run was scheduled to end.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@scheduledStartTime</td>
<td>The date and time that the run was scheduled to start.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>
**HiveCopyActivity**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>@status</td>
<td>The status of this object. This is a runtime slot. Possible values are: pending, waiting_on_dependencies, running, waiting_on_runner, successful, and failed.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@triesLeft</td>
<td>The number of attempted runs remaining before setting the status of this object to failed. This is a runtime slot.</td>
<td>Integer (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@waitingOn</td>
<td>A list of all objects that this object is waiting on before it can enter the RUNNING state.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields from SchedulableObject.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxActiveInstances</td>
<td>The maximum number of concurrent active instances of a component. For activities, setting this to 1 runs instances in strict chronological order. A value greater than 1 allows different instances of the activity to run concurrently and requires you to ensure your activity can tolerate concurrent execution.</td>
<td>Integer between 1 and 5</td>
<td>No</td>
</tr>
<tr>
<td>runsOn</td>
<td>The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.</td>
<td>Resource object reference</td>
<td>No</td>
</tr>
<tr>
<td>schedule</td>
<td>A schedule of the object. A common use is to specify a time schedule that correlates to the schedule for the object.</td>
<td>Schedule (p. 292) object reference</td>
<td>No</td>
</tr>
<tr>
<td>scheduleType</td>
<td>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of the interval or 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.</td>
<td>Allowed values are &quot;cron&quot; or &quot;timeseries&quot;. Defaults to &quot;timeseries&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>workerGroup</td>
<td>The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>healthStatus</td>
<td>The health status of the object, which reflects success or failure of the last instance that reached a terminated state. Values are: HEALTHY or ERROR.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>healthStatusFromInstanceId</td>
<td>The ID of the last object instance that reached a terminated state.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>
### 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.

```json
{
  "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"
      }
    }
  ]
}
```
{  
  "generatedScriptsPath": ":example-bucket/generatedScriptsPath",
  "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"
  },
  "generatedScriptsPath": ":example-bucket/generatedScriptsPath",
  "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": ":example-bucket/PigActivityOutput2",
  "dataFormat": {
    "ref": "MyOutputDataType2"
  },
  "type": "S3DataNode"
}
,
AWS Data Pipeline Developer Guide
PigActivity

},
"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"
},

API Version 2012-10-29
220


["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"
 },
"generatedScriptsPath": "s3://example-bucket/generatedScriptsPath",
"name": "MyPigActivity2"}
The content of pigTestScript.q is as follows.

```
B = LIMIT ${input1} $three; ${output1} = FOREACH B GENERATE Third, Fourth, Fifth, Sixth, Seventh, Eighth;",
```

The content of pigTestScript.q is as follows.

```
B = LIMIT ${input1} $three; ${output1} = FOREACH B GENERATE Third, Fourth, Fifth, Sixth, Seventh, Eighth;
```
# Syntax

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>The input data source.</td>
<td>Data node object reference</td>
<td>Yes</td>
</tr>
<tr>
<td>generatedScriptsPath</td>
<td>An Amazon S3 path to capture the Pig script that ran after all the expressions in it were evaluated, including staging information. This script is stored for historical, troubleshooting purposes.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>output</td>
<td>The location for the output.</td>
<td>Data node object reference</td>
<td>Yes</td>
</tr>
<tr>
<td>runsOn</td>
<td>The Amazon EMR cluster to run this activity.</td>
<td>EmrCluster (p. 250) object reference</td>
<td>Yes</td>
</tr>
<tr>
<td>script</td>
<td>The Pig script to run. You must specify either script or scriptUri.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>scriptUri</td>
<td>The location of the Pig script to run. For example, s3://script location. You must specify either scriptUri or script.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>scriptVariable</td>
<td>The arguments to pass to the Pig script. You can use scriptVariable with script or scriptUri.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>
### PigActivity

This object includes the following fields from the `Activity` object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>stage</td>
<td>Determines whether staging is enabled and allows your Pig script to have access to the staged-data tables, such as <code>${INPUT1}</code> and <code>${OUTPUT1}</code>.</td>
<td>Boolean</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields from `RunnableObject`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>dependsOn</td>
<td>One or more references to other Activities that must reach the FINISHED state before this activity will start.</td>
<td>Activity object reference</td>
<td>No</td>
</tr>
<tr>
<td>onFail</td>
<td>The SnsAlarm to use when the current instance fails.</td>
<td>SnsAlarm (p. 289) object reference</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>The SnsAlarm to use when the current instance succeeds.</td>
<td>SnsAlarm (p. 289) object reference</td>
<td>No</td>
</tr>
<tr>
<td>precondition</td>
<td>A condition that must be met before the object can run. To specify multiple conditions, add multiple precondition fields. The activity cannot run until all its conditions are met.</td>
<td>List of preconditions</td>
<td>No</td>
</tr>
<tr>
<td>schedule</td>
<td>A schedule of the object. A common use is to specify a time schedule that correlates to the schedule for the object. This slot overrides the <code>schedule</code> slot included from <code>SchedulableObject</code>, which is optional.</td>
<td>Schedule (p. 292) object reference</td>
<td>Yes</td>
</tr>
<tr>
<td>scheduleType</td>
<td>Specifies whether the pipeline component should be scheduled at the beginning of the interval or the end of the interval. <code>timeseries</code> means instances are scheduled at the end of each interval and <code>cron</code> means instances are scheduled at the beginning of each interval. The default value is <code>timeseries</code>.</td>
<td>Schedule (p. 292) object reference</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>@activeInstances</td>
<td>Record of the currently scheduled instance objects</td>
<td>Schedulable object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptStatus</td>
<td>The status most recently reported from the object.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@actualEndTime</td>
<td>The date and time that the scheduled run actually ended. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>@actualStartTime</td>
<td>The date and time that the scheduled run actually started. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptTimeout</td>
<td>The timeout time interval for an object attempt. If an attempt does not complete within the start time plus this time interval, AWS Data Pipeline marks the attempt as failed and your retry settings determine the next steps taken.</td>
<td>Time period; for example, &quot;1 hour&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@cascadeFailedOn</td>
<td>Description of which dependency the object failed on.</td>
<td>List of objects (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@componentParent</td>
<td>The component from which this instance is created.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorId</td>
<td>If the object failed, the error code. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorMessage</td>
<td>If the object failed, the error message. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorStackTrace</td>
<td>If the object failed, the error stack trace.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>failureAndRerunMode</td>
<td>Determines whether pipeline object failures and rerun commands cascade through pipeline object dependencies. For more information, see Cascading Failures and Reruns (p. 51).</td>
<td>String. Possible values are cascade and none.</td>
<td>No</td>
</tr>
<tr>
<td>@headAttempt</td>
<td>The latest attempt on the given instance.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@hostname</td>
<td>The host name of client that picked up the task attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>lateAfterTimeout</td>
<td>The time period in which the object run must start. If the object does not start within the scheduled start time plus this time interval, it is considered late.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>maximumRetries</td>
<td>The maximum number of times to retry the action. The default value is 2, which results in 3 tries total (1 original attempt plus 2 retries). The maximum value is 5 (6 total attempts).</td>
<td>Integer</td>
<td>No</td>
</tr>
<tr>
<td>onFail</td>
<td>An action to run when the current object fails.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onLateAction</td>
<td>The SnsAlarm to use when the object's run is late.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>An action to run when the current object succeeds.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
</tbody>
</table>
### PigActivity

This object includes the following fields from `SchedulableObject`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxActiveInstances</td>
<td>The maximum number of concurrent active instances of a component. For activities, setting this to 1 runs instances in strict chronological order. A value greater than 1 allows different instances of the activity to run concurrently and requires you to ensure your activity can tolerate concurrent execution.</td>
<td>Integer between 1 and 5</td>
<td>No</td>
</tr>
<tr>
<td>runsOn</td>
<td>The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.</td>
<td>Resource object reference</td>
<td>No</td>
</tr>
</tbody>
</table>
### Required

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>schedule</td>
<td>A schedule of the object. A common use is to specify a time schedule that correlates to the schedule for the object.</td>
<td>Schedule (p. 292) object reference</td>
<td>No</td>
</tr>
<tr>
<td>scheduleType</td>
<td>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of the interval or 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.</td>
<td>Allowed values are &quot;cron&quot; or &quot;timeseries&quot;. Defaults to &quot;timeseries&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>workerGroup</td>
<td>The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>healthStatus</td>
<td>The health status of the object, which reflects success or failure of the last instance that reached a terminated state. Values are: HEALTHY or ERROR.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>healthStatusFromInstanceId</td>
<td>The ID of the last object instance that reached a terminated state.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>healthStatusUpdatedTime</td>
<td>The last time at which the health status was updated.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>

### See Also

- ShellCommandActivity (p. 233)
- EmrActivity (p. 201)

### RedshiftCopyActivity

Copies data directly 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.187).

You can use SqlActivity (p. 239) to perform SQL queries on the data that you've loaded into Amazon Redshift.

### Example

The following is an example of this object type.

```json
{
  "id" : "S3ToRedshiftCopyActivity",
...}
```
"type" : "RedshiftCopyActivity",
"input" : { "ref": "MyS3DataNode" },
"output" : { "ref": "MyRedshiftDataNode" },
"insertMode" : "KEEP_EXISTING",
"schedule" : { "ref": "Hour" },
"runsOn" : { "ref": "MyEc2Resource" },
"commandOptions": ["EMPTYASNULL", "IGNOREBLANKLINES"]
}

For a tutorial, see Copy Data to Amazon Redshift Using AWS Data Pipeline (p. 131).

**Syntax**

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>The input data node. The data source can be Amazon S3, DynamoDB, or Amazon Redshift.</td>
<td>DataNode object reference</td>
<td>Yes</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------</td>
<td>----------</td>
</tr>
<tr>
<td>insertMode</td>
<td>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, and TRUNCATE. 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.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>output</td>
<td>The output data node. The output location can be Amazon S3 or Amazon Redshift.</td>
<td>DataNode object reference</td>
<td>Yes</td>
</tr>
<tr>
<td>transformSql</td>
<td>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. So transformSql must be run on the table named staging and the output schema of transformSql must match the final target table's schema.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>commandOptions</td>
<td>Takes COPY parameters to pass to the Amazon Redshift data node. For information about Amazon Redshift COPY parameters, see <a href="http://docs.aws.amazon.com/redshift/latest/dg/r_COPY.html#r_COPY-parameters">http://docs.aws.amazon.com/redshift/latest/dg/r_COPY.html#r_COPY-parameters</a>. If a data format is associated with the input or output data node, then the provided parameters are ignored.</td>
<td>List of Strings</td>
<td>No</td>
</tr>
<tr>
<td>queue</td>
<td>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.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>
This object includes the following fields from the Activity object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>dependsOn</td>
<td>One or more references to other Activities that must reach the FINISHED state before this activity will start.</td>
<td>Activity object reference</td>
<td>No</td>
</tr>
<tr>
<td>onFail</td>
<td>The SnsAlarm to use when the current instance fails.</td>
<td>SnsAlarm (p. 289) object reference</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>The SnsAlarm to use when the current instance succeeds.</td>
<td>SnsAlarm (p. 289) object reference</td>
<td>No</td>
</tr>
<tr>
<td>precondition</td>
<td>A condition that must be met before the object can run. To specify multiple conditions, add multiple precondition fields. The activity cannot run until all its conditions are met.</td>
<td>List of preconditions</td>
<td>No</td>
</tr>
<tr>
<td>schedule</td>
<td>A schedule of the object. A common use is to specify a time schedule that correlates to the schedule for the object.</td>
<td>Schedule (p. 292) object reference</td>
<td>Yes</td>
</tr>
<tr>
<td>scheduleType</td>
<td>Specifies whether the pipeline component should be scheduled at the beginning of the interval or the end of the interval. timeseries means instances are scheduled at the end of each interval and cron means instances are scheduled at the beginning of each interval. The default value is timeseries.</td>
<td>Schedule (p. 292) object reference</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields from RunnableObject.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>@activeInstances</td>
<td>Record of the currently scheduled instance objects</td>
<td>Schedulable object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptStatus</td>
<td>The status most recently reported from the object.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@actualEndTime</td>
<td>The date and time that the scheduled run actually ended. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@actualStartTime</td>
<td>The date and time that the scheduled run actually started. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>attemptTimeout</td>
<td>The timeout time interval for an object attempt. If an attempt does not complete within the start time plus this time interval, AWS Data Pipeline marks the attempt as failed and your retry settings determine the next steps taken.</td>
<td>Time period; for example, &quot;1 hour&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@cascadeFailedOn</td>
<td>Description of which dependency the object failed on.</td>
<td>List of objects (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@componentParent</td>
<td>The component from which this instance is created.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorId</td>
<td>If the object failed, the error code. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorMessage</td>
<td>If the object failed, the error message. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorStackTrace</td>
<td>If the object failed, the error stack trace.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>failureAndRerunMode</td>
<td>Determines whether pipeline object failures and rerun commands cascade through pipeline object dependencies. For more information, see Cascading Failures and Reruns (p. 51).</td>
<td>String. Possible values are cascade and none.</td>
<td>No</td>
</tr>
<tr>
<td>@headAttempt</td>
<td>The latest attempt on the given instance.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@hostname</td>
<td>The host name of client that picked up the task attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>lateAfterTimeout</td>
<td>The time period in which the object run must start. If the object does not start within the scheduled start time plus this time interval, it is considered late.</td>
<td>Time period; for example, &quot;1 hour&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>maximumRetries</td>
<td>The maximum number of times to retry the action. The default value is 2, which results in 3 tries total (1 original attempt plus 2 retries). The maximum value is 5 (6 total attempts).</td>
<td>Integer</td>
<td>No</td>
</tr>
<tr>
<td>onFail</td>
<td>An action to run when the current object fails.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onLateAction</td>
<td>The SnsAlarm to use when the object’s run is late.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>An action to run when the current object succeeds.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>@reportProgressTime</td>
<td>The last time that Task Runner, or other code that is processing the tasks, called the ReportTaskProgress API.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>
This object includes the following fields from `SchedulableObject`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxActiveInstances</td>
<td>The maximum number of concurrent active instances of a component. For activities, setting this to 1 runs instances in strict chronological order. A value greater than 1 allows different instances of the activity to run concurrently and requires you to ensure your activity can tolerate concurrent execution.</td>
<td>Integer between 1 and 5</td>
<td>No</td>
</tr>
<tr>
<td>runsOn</td>
<td>The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.</td>
<td>Resource object reference</td>
<td>No</td>
</tr>
<tr>
<td>schedule</td>
<td>A schedule of the object. A common use is to specify a time schedule that correlates to the schedule for the object.</td>
<td>Schedule (p. 292) object reference</td>
<td>No</td>
</tr>
</tbody>
</table>
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 to 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.

```json
{
    "id" : "CreateDirectory",
    "type" : "ShellCommandActivity",
    "command" : "mkdir new-directory"
}
```
Syntax

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>name for an object in a pipeline definition, AWS Data Pipeline automatically</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>duplicates the value of id.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>as Pipeline, Component, Instance, or Attempt.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>types.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>command</td>
<td>The command to run. This value and any associated parameters must function</td>
<td>String</td>
<td>Conditional</td>
</tr>
<tr>
<td></td>
<td>in the environment from which you are running the Task Runner.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>input</td>
<td>The input data source.</td>
<td>Data node object reference</td>
<td>No</td>
</tr>
<tr>
<td>output</td>
<td>The location for the output.</td>
<td>Data node object reference</td>
<td>No</td>
</tr>
<tr>
<td>scriptArgument</td>
<td>A list of arguments to pass to the shell script.</td>
<td>List of strings</td>
<td>No</td>
</tr>
<tr>
<td>scriptUri</td>
<td>An Amazon S3 URI path for a file to download and run as a shell command.</td>
<td>A valid S3 URI</td>
<td>Conditional</td>
</tr>
<tr>
<td></td>
<td>Only one scriptUri or command field should be present. scriptUri cannot</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>use parameters, use command instead.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>stage</td>
<td>Determines whether staging is enabled and allows your shell commands to</td>
<td>Boolean</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>have access to the staged-data variables, such as ${INPUT1_STAGING_DIR} and</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>${OUTPUT1_STAGING_DIR}.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**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.

Type: **String**. For example, `s3://examples-bucket/script_stderr`.

**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.

Type: **String**. For example, `s3://examples-bucket/script_stdout`.

**Note**
You must specify a `command` value or a `scriptUri` value, but you are not required to specify both.

This object includes the following fields from the Activity object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dependson</code></td>
<td>One or more references to other Activities that must reach the FINISHED state before this activity will start.</td>
<td>Activity object reference</td>
<td>No</td>
</tr>
<tr>
<td><code>onFail</code></td>
<td>The SnsAlarm to use when the current instance fails.</td>
<td>SnsAlarm (p. 289) object reference</td>
<td>No</td>
</tr>
<tr>
<td><code>onSuccess</code></td>
<td>The SnsAlarm to use when the current instance succeeds.</td>
<td>SnsAlarm (p. 289) object reference</td>
<td>No</td>
</tr>
<tr>
<td><code>precondition</code></td>
<td>A condition that must be met before the object can run. To specify multiple conditions, add multiple <code>precondition</code> fields. The activity cannot run until all its conditions are met.</td>
<td>List of preconditions</td>
<td>No</td>
</tr>
<tr>
<td><code>schedule</code></td>
<td>A schedule of the object. A common use is to specify a time schedule that correlates to the schedule for the object.</td>
<td>Schedule (p. 292) object reference</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>This slot overrides the schedule slot included from SchedulableObject, which is optional.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>scheduleType</code></td>
<td>Specifies whether the pipeline component should be scheduled at the beginning of the interval or the end of the interval. <code>timeseries</code> means instances are scheduled at the end of each interval and <code>cron</code> means instances are scheduled at the beginning of each interval. The default value is <code>timeseries</code>.</td>
<td>Schedule (p. 292) object reference</td>
<td>No</td>
</tr>
</tbody>
</table>
This object includes the following fields from `RunnableObject`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>@activeInstances</td>
<td>Record of the currently scheduled instance objects</td>
<td>Schedulable object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptStatus</td>
<td>The status most recently reported from the object.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@actualEndTime</td>
<td>The date and time that the scheduled run actually ended. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@actualStartTime</td>
<td>The date and time that the scheduled run actually started. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptTimeout</td>
<td>The timeout time interval for an object attempt. If an attempt does not complete within the start time plus this time interval, AWS Data Pipeline marks the attempt as failed and your retry settings determine the next steps taken.</td>
<td>Time period; for example, “1 hour”.</td>
<td>No</td>
</tr>
<tr>
<td>@cascadeFailedOn</td>
<td>Description of which dependency the object failed on.</td>
<td>List of objects (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@componentParent</td>
<td>The component from which this instance is created.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorId</td>
<td>If the object failed, the error code. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorMessage</td>
<td>If the object failed, the error message. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorStackTrace</td>
<td>If the object failed, the error stack trace.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>failureAndRerunMode</td>
<td>Determines whether pipeline object failures and rerun commands cascade through pipeline object dependencies. For more information, see Cascading Failures and Reruns (p. 51).</td>
<td>String. Possible values are <code>cascade</code> and <code>none</code>.</td>
<td>No</td>
</tr>
<tr>
<td>@headAttempt</td>
<td>The latest attempt on the given instance.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@hostname</td>
<td>The host name of client that picked up the task attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>lateAfterTimeout</td>
<td>The time period in which the object run must start. If the object does not start within the scheduled start time plus this time interval, it is considered late.</td>
<td>Time period; for example, “1 hour”. The minimum value is “15 minutes”.</td>
<td>No</td>
</tr>
<tr>
<td>maximumRetries</td>
<td>The maximum number of times to retry the action. The default value is 2, which results in 3 tries total (1 original attempt plus 2 retries). The maximum value is 5 (6 total attempts).</td>
<td>Integer</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>onFail</td>
<td>An action to run when the current object fails.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onLateAction</td>
<td>The SnsAlarm to use when the object’s run is late.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>An action to run when the current object succeeds.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>@reportProgressTime</td>
<td>The last time that Task Runner, or other code that is processing the tasks, called the ReportTaskProgress API.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>reportProgressTimeout</td>
<td>The time period for successive calls from Task Runner to the ReportTaskProgress API. If Task Runner, or other code that is processing the tasks, does not report progress within the specified period, the object attempt can be retried.</td>
<td>Time period; for example, “1 hour”. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@resource</td>
<td>The resource instance on which the given activity/precondition attempt is being run.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>retryDelay</td>
<td>The timeout duration between two retry attempts. The default is 10 minutes.</td>
<td>Period. Minimum is &quot;1 second&quot;.</td>
<td>Yes</td>
</tr>
<tr>
<td>@scheduledEndTime</td>
<td>The date and time that the run was scheduled to end.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@scheduledStartTime</td>
<td>The date and time that the run was scheduled to start.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@status</td>
<td>The status of this object. This is a runtime slot. Possible values are: pending, waiting_on_dependencies, running, waiting_on_runner, successful, and failed.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@triesLeft</td>
<td>The number of attempted runs remaining before setting the status of this object to failed. This is a runtime slot.</td>
<td>Integer (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@waitingOn</td>
<td>A list of all objects that this object is waiting on before it can enter the RUNNING state.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>
This object includes the following fields from `SchedulableObject`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>maxActiveInstances</code></td>
<td>The maximum number of concurrent active instances of a component. For activities, setting this to 1 runs instances in strict chronological order. A value greater than 1 allows different instances of the activity to run concurrently and requires you to ensure your activity can tolerate concurrent execution.</td>
<td>Integer between 1 and 5</td>
<td>No</td>
</tr>
<tr>
<td><code>runsOn</code></td>
<td>The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.</td>
<td>Resource object reference</td>
<td>No</td>
</tr>
<tr>
<td><code>schedule</code></td>
<td>A schedule of the object. A common use is to specify a time schedule that correlates to the schedule for the object.</td>
<td>Schedule (p. 292) object reference</td>
<td>No</td>
</tr>
<tr>
<td><code>scheduleType</code></td>
<td>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of the interval or 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.</td>
<td>Allowed values are &quot;cron&quot; or &quot;timeseries&quot;. Defaults to &quot;timeseries&quot;.</td>
<td>No</td>
</tr>
<tr>
<td><code>workerGroup</code></td>
<td>The worker group. This is used for routing tasks. If you provide a <code>runsOn</code> value and <code>workerGroup</code> exists, <code>workerGroup</code> is ignored.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td><code>healthStatus</code></td>
<td>The health status of the object, which reflects success or failure of the last instance that reached a terminated state. Values are: HEALTHY or ERROR.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td><code>healthStatusFromInstanceId</code></td>
<td>The ID of the last object instance that reached a terminated state.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td><code>healthStatusUpdatedTime</code></td>
<td>The last time at which the health status was updated.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>

**See Also**

- [CopyActivity](#)
- [EmrActivity](#)
SqlActivity

Runs a SQL query on a database. You specify the input table where the SQL query is run and the output table where the results are stored. If the output table doesn't exist, this operation creates a new table with that name.

Example

The following is an example of this object type.

```
{
  "id" : "MySqlActivity",
  "type" : "SqlActivity",
  "input" : { "ref": "MyInputDataNode" },
  "output" : { "ref": "MyOutputDataNode" },
  "database" : { "ref": "MyDatabase" },
  "script" : "insert into AnalyticsTable (select (cast(requestEndTime as bigint) 
  - cast(requestBeginTime as bigint)) as requestTime, hostname from StructuredLogs
  where hostname LIKE '%.domain.sfx');",
  "schedule" : { "ref": "Hour" },
  "queue" : "priority"
}
```

Syntax

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>
This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>database</td>
<td>The database.</td>
<td>Database object reference</td>
<td>Yes</td>
</tr>
<tr>
<td>script</td>
<td>The SQL script to run. For example:</td>
<td>String</td>
<td>Conditional</td>
</tr>
<tr>
<td></td>
<td><code>insert into output select * from input where lastModified in range (?, ?)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>You must specify <code>script</code> or <code>scriptUri</code>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>When the <code>script</code> is stored in Amazon S3, the <code>script</code> is not evaluated as</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>an expression. In that situation, <code>scriptArguments</code> are helpful.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scriptArgument</td>
<td>A list of variables for the script. For example:</td>
<td>List of strings</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td><code>#{format(@scheduledStartTime, &quot;YY-MM-DD HH:MM:SS&quot;)}</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>#{format(plusPeriod(@scheduledStartTime, &quot;1 day&quot;), &quot;YY-MM-DD HH:MM:SS&quot;)}</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Alternatively, you can put expressions directly into the <code>script</code> field.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>scriptArguments</code> are helpful when the <code>script</code> is stored in Amazon S3.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scriptUri</td>
<td>The location of the SQL script to run. For example, <code>s3://script_location</code>.</td>
<td>String</td>
<td>Conditional</td>
</tr>
<tr>
<td>queue</td>
<td>Corresponds to the <code>query_group</code> setting in Amazon Redshift, which allows</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>you to assign and prioritize concurrent activities based on their</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>placement in queues. Amazon Redshift limits the number of simultaneous</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>connections to 15. For more information, see <code>Assigning Queries to Queues</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This object includes the following fields from the `Activity` object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>dependsOn</td>
<td>One or more references to other Activities that must reach the FINISHED</td>
<td>Activity object reference</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>state before this activity will start.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>onFail</td>
<td>The SnsAlarm to use when the current instance fails.</td>
<td>SnsAlarm (p. 289) object</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>The SnsAlarm to use when the current instance succeeds.</td>
<td>SnsAlarm (p. 289) object</td>
<td>No</td>
</tr>
</tbody>
</table>

API Version 2012-10-29
240
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>precondition</td>
<td>A condition that must be met before the object can run. To specify multiple conditions, add multiple precondition fields. The activity cannot run until all its conditions are met.</td>
<td>List of preconditions</td>
<td>No</td>
</tr>
<tr>
<td>schedule</td>
<td>A schedule of the object. A common use is to specify a time schedule that correlates to the schedule for the object. This slot overrides the schedule slot included from SchedulableObject, which is optional.</td>
<td>Schedule (p. 292) object reference</td>
<td>Yes</td>
</tr>
<tr>
<td>scheduleType</td>
<td>Specifies whether the pipeline component should be scheduled at the beginning of the interval or the end of the interval. timeseries means instances are scheduled at the end of each interval and cron means instances are scheduled at the beginning of each interval. The default value is timeseries.</td>
<td>Schedule (p. 292) object reference</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields from RunnableObject.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>@activeInstances</td>
<td>Record of the currently scheduled instance objects</td>
<td>Schedulable object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptStatus</td>
<td>The status most recently reported from the object.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@actualEndTime</td>
<td>The date and time that the scheduled run actually ended. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@actualStartTime</td>
<td>The date and time that the scheduled run actually started. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptTimeout</td>
<td>The timeout time interval for an object attempt. If an attempt does not complete within the start time plus this time interval, AWS Data Pipeline marks the attempt as failed and your retry settings determine the next steps taken.</td>
<td>Time period; for example, &quot;1 hour&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@cascadeFailedOn</td>
<td>Description of which dependency the object failed on.</td>
<td>List of objects (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@componentParent</td>
<td>The component from which this instance is created.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorId</td>
<td>If the object failed, the error code. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorMessage</td>
<td>If the object failed, the error message. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>errorStackTrace</td>
<td>If the object failed, the error stack trace.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>failureAndRerunMode</td>
<td>Determines whether pipeline object failures and rerun commands cascade through pipeline object dependencies. For more information, see Cascading Failures and Reruns (p. 51).</td>
<td>String. Possible values are cascade and none.</td>
<td>No</td>
</tr>
<tr>
<td>@headAttempt</td>
<td>The latest attempt on the given instance.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@hostname</td>
<td>The host name of client that picked up the task attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>lateAfterTimeout</td>
<td>The time period in which the object run must start. If the object does not start within the scheduled start time plus this time interval, it is considered late.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>maximumRetries</td>
<td>The maximum number of times to retry the action. The default value is 2, which results in 3 tries total (1 original attempt plus 2 retries). The maximum value is 5 (6 total attempts).</td>
<td>Integer</td>
<td>No</td>
</tr>
<tr>
<td>onFail</td>
<td>An action to run when the current object fails.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onLateAction</td>
<td>The SnsAlarm to use when the object's run is late.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>An action to run when the current object succeeds.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>@reportProgressTime</td>
<td>The last time that Task Runner, or other code that is processing the tasks, called the ReportTaskProgress API.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>reportProgressTimeout</td>
<td>The time period for successive calls from Task Runner to the ReportTaskProgress API. If Task Runner, or other code that is processing the tasks, does not report progress within the specified period, the object attempt can be retried.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@resource</td>
<td>The resource instance on which the given activity/precondition attempt is being run.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>retryDelay</td>
<td>The timeout duration between two retry attempts. The default is 10 minutes.</td>
<td>Period. Minimum is &quot;1 second&quot;.</td>
<td>Yes</td>
</tr>
<tr>
<td>@scheduledEndTime</td>
<td>The date and time that the run was scheduled to end.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@scheduledStartTime</td>
<td>The date and time that the run was scheduled to start.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>@status</td>
<td>The status of this object. This is a runtime slot. Possible values are: pending, waiting_on_dependencies, running, waiting_on_runner, successful, and failed.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@triesLeft</td>
<td>The number of attempted runs remaining before setting the status of this object to failed. This is a runtime slot.</td>
<td>Integer (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@waitingOn</td>
<td>A list of all objects that this object is waiting on before it can enter the RUNNING state.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields from SchedulableObject.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxActiveInstances</td>
<td>The maximum number of concurrent active instances of a component. For activities, setting this to 1 runs instances in strict chronological order. A value greater than 1 allows different instances of the activity to run concurrently and requires you to ensure your activity can tolerate concurrent execution.</td>
<td>Integer between 1 and 5</td>
<td>No</td>
</tr>
<tr>
<td>runsOn</td>
<td>The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.</td>
<td>Resource object reference</td>
<td>No</td>
</tr>
<tr>
<td>schedule</td>
<td>A schedule of the object. A common use is to specify a time schedule that correlates to the schedule for the object.</td>
<td>Schedule (p. 292) object reference</td>
<td>No</td>
</tr>
<tr>
<td>scheduleType</td>
<td>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of the interval or 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.</td>
<td>Allowed values are &quot;cron&quot; or &quot;timeseries&quot;. Defaults to &quot;timeseries&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>workerGroup</td>
<td>The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>healthStatus</td>
<td>The health status of the object, which reflects success or failure of the last instance that reached a terminated state. Values are: HEALTHY or ERROR.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>healthStatusFromInstanceId</td>
<td>The ID of the last object instance that reached a terminated state.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>
Resources

The following are Data Pipeline Resources:

Topics
- Ec2Resource (p. 244)
- EmrCluster (p. 250)

Ec2Resource

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

Example

The following is an example of this object type that launches an EC2 instance into EC2-Classic or a default VPC, with some optional fields set.

```json
{
    "id" : "MyEC2Resource",
    "type" : "Ec2Resource",
    "actionOnTaskFailure" : "terminate",
    "actionOnResourceFailure" : "retryAll",
    "maximumRetries" : "1",
    "role" : "test-role",
    "resourceRole" : "test-role",
    "instanceType" : "m1.medium",
    "securityGroups" : [ "test-group",
        "default"
    ],
    "keyPair" : "my-key-pair"
}
```

The following is an example of the object type that launches an EC2 instance into a nondefault VPC, with some optional fields set.

```json
{
    "id" : "MyEC2Resource",
    "type" : "Ec2Resource",
    "actionOnTaskFailure" : "terminate",
    "actionOnResourceFailure" : "retryAll",
    "maximumRetries" : "1",
    "role" : "test-role",
    "resourceRole" : "test-role",
    "instanceType" : "m1.medium",
    "securityGroupIds" : [ "sg-12345678",
        ]
}
```
Ec2Resource can run in the same region with its working data set, even a region different than AWS Data Pipeline. For more information, see Using a Pipeline with Resources in Multiple Regions (p. 50).

**Syntax**

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>associatePublicIpAddress</td>
<td>Indicates whether to assign a public IP address to an instance. (An instance in a VPC can't access Amazon S3 unless it has a public IP address or a network address translation (NAT) instance with proper routing configuration.) If the instance is in EC2-Classic or a default VPC, the default value is true. Otherwise, the default value is false.</td>
<td>Boolean</td>
<td>No</td>
</tr>
<tr>
<td>imageId</td>
<td>The AMI version to use for the EC2 instances. For more information, see Amazon Machine Images (AMIs).</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>
### Instance Type

The type of EC2 instance to use for the resource pool. The default value is `m1.small`. The values currently supported are:

- `c1.medium`
- `c1.xlarge`
- `c3.2xlarge`
- `c3.4xlarge`
- `c3.8xlarge`
- `c3.xlarge`
- `cc1.4xlarge`
- `cc2.8xlarge`
- `cg1.4xlarge`
- `cr1.8xlarge`
- `g2.2xlarge`
- `hi1.4xlarge`
- `hs1.8xlarge`
- `i2.2xlarge`
- `i2.4xlarge`
- `i2.8xlarge`
- `i2.xlarge`
- `m1.large`
- `m1.medium`
- `m1.small`
- `m1.xlarge`
- `m2.2xlarge`
- `m2.4xlarge`
- `m2.xlarge`
- `m3.2xlarge`
- `m3.xlarge`
- `t1.micro`

### Key Pair

**Important**

The Amazon EC2 key pair is required to log onto the EC2 instance. The default action is not to attach a key pair to the EC2 instance.

### Log URI

The Amazon S3 destination path to back up Task Runner logs from `Ec2Resource/EmrCluster` resource.

### Region

A region code to specify that the resource should run in a different region. For more information, see Using a Pipeline with Resources in Multiple Regions (p. 50).

### Resource Role

The IAM role to use to control the resources that the EC2 instance can access.

### Role

The IAM role to use to create the EC2 instance.

### Security Group IDs

The IDs of one or more security groups to use for the instances in the resource pool. By default, Amazon EC2 uses the default security group. The maximum number of security groups is 10.

### Security Groups

The names of one or more security groups to use for the instances in the resource pool. By default, Amazon EC2 uses the default security group. The maximum number of security groups is 10. If your instance is in a nondefault VPC, you must use `securityGroupIds` to specify security groups.

### Subnet ID

The ID of the subnet to launch the cluster into.
This object includes the following fields from the Resource object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>actionOnResourceFailure</code></td>
<td>Action to take when the resource fails.</td>
<td>String: <code>retryall</code> (retry all inputs) or <code>retrynone</code> (retry nothing)</td>
<td>No</td>
</tr>
<tr>
<td><code>actionOnTaskFailure</code></td>
<td>Action to take when the task associated with this resource fails.</td>
<td>String: <code>continue</code> (do not terminate the cluster) or <code>terminate</code></td>
<td>No</td>
</tr>
<tr>
<td><code>@failureReason</code></td>
<td>The reason for the failure to create the resource.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td><code>logUri</code></td>
<td>The Amazon S3 destination path to back up Task Runner logs from Ec2Resource/EmrCluster resource.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td><code>region</code></td>
<td>The AWS region in which the resource will launch. The default value is the region in which you run AWS Data Pipeline.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td><code>@resourceCreationTime</code></td>
<td>The time when this resource was created.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td><code>@resourceId</code></td>
<td>The unique identifier for the resource.</td>
<td>Period (read-only)</td>
<td>Yes</td>
</tr>
<tr>
<td><code>@resourceStatus</code></td>
<td>The current status of the resource, such as WAITING_ON_DEPENDENCIES, creating, shutting_down, running, failed, timed_out, cancelled, or paused.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td><code>terminateAfter</code></td>
<td>The amount of time to wait before terminating the resource.</td>
<td>Time period; for example, “1 hour”.</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields from RunnableObject.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>@activeInstances</code></td>
<td>Record of the currently scheduled instance objects</td>
<td>Schedulable object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td><code>attemptStatus</code></td>
<td>The status most recently reported from the object.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td><code>@actualEndTime</code></td>
<td>The date and time that the scheduled run actually ended. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td><code>@actualStartTime</code></td>
<td>The date and time that the scheduled run actually started. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td><code>attemptTimeout</code></td>
<td>The timeout time interval for an object attempt. If an attempt does not complete within the start time plus this time interval, AWS Data Pipeline marks the attempt as failed and your retry settings determine the next steps taken.</td>
<td>Time period; for example, “1 hour”.</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>@cascadeFailedOn</td>
<td>Description of which dependency the object failed on.</td>
<td>List of objects (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@componentParent</td>
<td>The component from which this instance is created.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorId</td>
<td>If the object failed, the error code. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorMessage</td>
<td>If the object failed, the error message. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorStackTrace</td>
<td>If the object failed, the error stack trace.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>failureAndRerunMode</td>
<td>Determines whether pipeline object failures and rerun commands cascade through pipeline object dependencies. For more information, see Cascading Failures and Reruns (p. 51).</td>
<td>String, Possible values are cascade and none.</td>
<td>No</td>
</tr>
<tr>
<td>@headAttempt</td>
<td>The latest attempt on the given instance.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@hostname</td>
<td>The host name of client that picked up the task attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>lateAfterTimeout</td>
<td>The time period in which the object run must start. If the object does not start within the scheduled start time plus this time interval, it is considered late.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>maximumRetries</td>
<td>The maximum number of times to retry the action. The default value is 2, which results in 3 tries total (1 original attempt plus 2 retries). The maximum value is 5 (6 total attempts).</td>
<td>Integer</td>
<td>No</td>
</tr>
<tr>
<td>onLateAction</td>
<td>The SnsAlarm to use when the object's run is late.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>@reportProgressTime</td>
<td>The last time that Task Runner, or other code that is processing the tasks, called the ReportTaskProgress API.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>reportProgressTimeout</td>
<td>The time period for successive calls from Task Runner to the ReportTaskProgress API. If Task Runner, or other code that is processing the tasks, does not report progress within the specified period, the object attempt can be retried.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@resource</td>
<td>The resource instance on which the given activity/precondition attempt is being run.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>retryDelay</td>
<td>The timeout duration between two retry attempts. The default is 10 minutes.</td>
<td>Period. Minimum is &quot;1 second&quot;.</td>
<td>Yes</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>@scheduledEndTime</td>
<td>The date and time that the run was scheduled to end.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@scheduledStartTime</td>
<td>The date and time that the run was scheduled to start.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@status</td>
<td>The status of this object. This is a runtime slot. Possible values are: pending, waiting_on_dependencies, running, waiting_on_runner, successful, and failed.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@triesLeft</td>
<td>The number of attempted runs remaining before setting the status of this object to failed. This is a runtime slot.</td>
<td>Integer (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@waitingOn</td>
<td>A list of all objects that this object is waiting on before it can enter the RUNNING state.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields from SchedulableObject.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxActiveInstances</td>
<td>The maximum number of concurrent active instances of a component. For activities, setting this to 1 runs instances in strict chronological order. A value greater than 1 allows different instances of the activity to run concurrently and requires you to ensure your activity can tolerate concurrent execution.</td>
<td>Integer between 1 and 5</td>
<td>No</td>
</tr>
<tr>
<td>schedule</td>
<td>A schedule of the object. A common use is to specify a time schedule that correlates to the schedule for the object.</td>
<td>Schedule (p. 292) object reference</td>
<td>No</td>
</tr>
<tr>
<td>scheduleType</td>
<td>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of the interval or 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. Allowed values are &quot;cron&quot; or &quot;timeseries&quot;. Defaults to &quot;timeseries&quot;.</td>
<td>Allowed values are &quot;cron&quot; or &quot;timeseries&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>healthStatus</td>
<td>The health status of the object, which reflects success or failure of the last instance that reached a terminated state. Values are: HEALTHY or ERROR.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>healthStatusFromInstanceId</td>
<td>The ID of the last object instance that reached a terminated state.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>healthStatusUpdatedTime</td>
<td>The last time at which the health status was updated.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>
EmrCluster

Represents the configuration of an Amazon EMR cluster. This object is used by EmrActivity (p. 201) to launch a cluster.

Example

The following is an example of this object type. It launches an Amazon EMR cluster using AMI version 1.0 and Hadoop 0.20.

```json
{
    "id" : "MyEmrCluster",
    "type" : "EmrCluster",
    "hadoopVersion" : "0.20",
    "keypair" : "my-key-pair",
    "masterInstanceType" : "m1.xlarge",
    "coreInstanceType" : "m1.small",
    "coreInstanceCount" : "10",
    "taskInstanceType" : "m1.small",
    "taskInstanceCount": "10",
    "bootstrapAction" : ["s3://elasticmapreduce/libs/ba/configure-hadoop,arg1,arg2,arg3","s3://elasticmapreduce/libs/ba/configure-other-stuff,arg1,arg2"]
}
```

EmrCluster can run in the same region with its working data set, even a region different than AWS Data Pipeline. For more information, see Using a Pipeline with Resources in Multiple Regions (p. 50).

EmrCluster provides the supportedProducts field that installs third-party software on an Amazon EMR cluster, for example installing a custom distribution of Hadoop. 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 the EmrCluster to create a custom MapR M3 edition cluster with Karmasphere Analytics installed and run an EmrActivity on it.

```json
{
    "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": "m1.medium",
    "taskInstanceType": "m1.medium"
}
```
Syntax

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>amiVersion</td>
<td>The Amazon Machine Image (AMI) version to use by Amazon EMR to install the cluster nodes. The default value is “2.2.4”. For more information, see <a href="#">AMI Versions Supported in Amazon EMR</a>.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>bootstrapAction</td>
<td>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.</td>
<td>String array</td>
<td>No</td>
</tr>
<tr>
<td>coreInstanceCount</td>
<td>The number of core nodes to use for the cluster. The default value is 1.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>coreInstanceType</td>
<td>The type of EC2 instance to use for core nodes. The default value is ml.small.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>emrLogUri</td>
<td>The Amazon S3 destination path to write the Amazon EMR debugging logs. You must set this value along with enableDebugging set to true for the Debug button to work in the Amazon EMR console.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>enableDebugging</td>
<td>Enables debugging on the Amazon EMR cluster.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>hadoopVersion</td>
<td>The version of Hadoop to use in the cluster. The default value is 0.20. For more information about the Hadoop versions supported by Amazon EMR, see Supported Hadoop Versions.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>installHive</td>
<td>The Hive version or versions to load. This can be a Hive version number or &quot;latest&quot; to load the latest version. When you specify more than one Hive version, separate the versions with a comma.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>keyPair</td>
<td>The Amazon EC2 key pair to use to log onto the master node of the cluster. The default action is not to attach a key pair to the cluster.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>masterInstanceType</td>
<td>The type of EC2 instance to use for the master node. The default value is <code>m1.small</code>.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>region</td>
<td>A region code to specify that the resource should run in a different region. For more information, see Using a Pipeline with Resources in Multiple Regions (p. 50).</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>subnetId</td>
<td>The ID of the subnet to launch the cluster into.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>supportedProducts</td>
<td>A parameter that installs third-party software on an Amazon EMR cluster, for example installing a third-party distribution of Hadoop.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>taskInstanceBidPrice</td>
<td>The maximum dollar amount for your Spot Instance bid and is a decimal value between 0 and 20.00 exclusive. Setting this value enables Spot Instances for the EMR cluster task nodes.</td>
<td>Decimal</td>
<td>No</td>
</tr>
<tr>
<td>taskInstanceCount</td>
<td>The number of task nodes to use for the cluster. The default value is 1.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>taskInstanceType</td>
<td>The type of EC2 instance to use for task nodes.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>pigVersion</td>
<td>The version or versions of Pig to load. This can be a Pig version number or &quot;latest&quot; to load the latest version. When you specify more than one Pig version, separate the versions with a comma.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>
This object includes the following fields from the `Resource` object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>actionOnResourceFailure</td>
<td>Action to take when the resource fails.</td>
<td>String: <code>retryall</code> (retry all inputs) or <code>retrynone</code> (retry nothing)</td>
<td>No</td>
</tr>
<tr>
<td>actionOnTaskFailure</td>
<td>Action to take when the task associated with this resource fails.</td>
<td>String: <code>continue</code> (do not terminate the cluster) or <code>terminate</code></td>
<td>No</td>
</tr>
<tr>
<td>@failureReason</td>
<td>The reason for the failure to create the resource.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>logUri</td>
<td>The Amazon S3 destination path to back up Task Runner logs from Ec2Resource/EmrCluster resource.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>region</td>
<td>The AWS region in which the resource will launch. The default value is the region in which you run AWS Data Pipeline.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@resourceCreationTime</td>
<td>The time when this resource was created.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@resourceId</td>
<td>The unique identifier for the resource.</td>
<td>Period (read-only)</td>
<td>Yes</td>
</tr>
<tr>
<td>@resourceStatus</td>
<td>The current status of the resource, such as <code>WAITING_ON_DEPENDENCIES</code>, creating, shutting_down, running, failed, timed_out, cancelled, or paused.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>terminateAfter</td>
<td>The amount of time to wait before terminating the resource.</td>
<td>Time period; for example, &quot;1 hour&quot;.</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields from `RunnableObject`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>@activeInstances</td>
<td>Record of the currently scheduled instance objects</td>
<td>Schedulable object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptStatus</td>
<td>The status most recently reported from the object.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@actualEndTime</td>
<td>The date and time that the scheduled run actually ended. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@actualStartTime</td>
<td>The date and time that the scheduled run actually started. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptTimeout</td>
<td>The timeout time interval for an object attempt. If an attempt does not complete within the start time plus this time interval, AWS Data Pipeline marks the attempt as failed and your retry settings determine the next steps taken.</td>
<td>Time period; for example, &quot;1 hour&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>@cascadeFailedOn</td>
<td>Description of which dependency the object failed on.</td>
<td>List of objects (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@componentParent</td>
<td>The component from which this instance is created.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorId</td>
<td>If the object failed, the error code. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorMessage</td>
<td>If the object failed, the error message. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorStackTrace</td>
<td>If the object failed, the error stack trace.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>failureAndRerunMode</td>
<td>Determines whether pipeline object failures and rerun commands cascade through pipeline object dependencies. For more information, see Cascading Failures and Reruns (p. 51).</td>
<td>String, Possible values are cascade and none.</td>
<td>No</td>
</tr>
<tr>
<td>@headAttempt</td>
<td>The latest attempt on the given instance.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@hostname</td>
<td>The host name of client that picked up the task attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>lateAfterTimeout</td>
<td>The time period in which the object run must start. If the object does not start within the scheduled start time plus this time interval, it is considered late.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>maximumRetries</td>
<td>The maximum number of times to retry the action. The default value is 2, which results in 3 tries total (1 original attempt plus 2 retries). The maximum value is 5 (6 total attempts).</td>
<td>Integer</td>
<td>No</td>
</tr>
<tr>
<td>onLateAction</td>
<td>The SnsAlarm to use when the object’s run is late.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>@reportProgressTime</td>
<td>The last time that Task Runner, or other code that is processing the tasks, called the ReportTaskProgress API.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>reportProgressTimeout</td>
<td>The time period for successive calls from Task Runner to the ReportTaskProgress API. If Task Runner, or other code that is processing the tasks, does not report progress within the specified period, the object attempt can be retried.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@resource</td>
<td>The resource instance on which the given activity/precondition attempt is being run.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>retryDelay</td>
<td>The timeout duration between two retry attempts. The default is 10 minutes.</td>
<td>Period. Minimum is &quot;1 second&quot;.</td>
<td>Yes</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>@scheduledEndTime</td>
<td>The date and time that the run was scheduled to end.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@scheduledStartTime</td>
<td>The date and time that the run was scheduled to start.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@status</td>
<td>The status of this object. This is a runtime slot. Possible values are: pending, waiting_on_dependencies, running, waiting_on_runner, successful, and failed.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@triesLeft</td>
<td>The number of attempted runs remaining before setting the status of this object to failed. This is a runtime slot.</td>
<td>Integer (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@waitingOn</td>
<td>A list of all objects that this object is waiting on before it can enter the RUNNING state.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields from SchedulableObject.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxActiveInstances</td>
<td>The maximum number of concurrent active instances of a component. For activities, setting this to 1 runs instances in strict chronological order. A value greater than 1 allows different instances of the activity to run concurrently and requires you to ensure your activity can tolerate concurrent execution.</td>
<td>Integer between 1 and 5</td>
<td>No</td>
</tr>
<tr>
<td>schedule</td>
<td>A schedule of the object. A common use is to specify a time schedule that correlates to the schedule for the object.</td>
<td>Schedule (p. 292) object reference</td>
<td>No</td>
</tr>
<tr>
<td>scheduleType</td>
<td>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of the interval or 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.</td>
<td>Allowed values are &quot;cron&quot; or &quot;timeseries&quot;. Defaults to &quot;timeseries&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>healthStatus</td>
<td>The health status of the object, which reflects success or failure of the last instance that reached a terminated state. Values are: HEALTHY or ERROR.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>healthStatusFromInstanceId</td>
<td>The ID of the last object instance that reached a terminated state.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>healthStatusUpdatedTime</td>
<td>The last time at which the health status was updated.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>
Preconditions

The following are Data Pipeline Preconditions:

Topics

- DynamoDBDataExists (p. 256)
- DynamoDBTableExists (p. 259)
- Exists (p. 262)
- S3KeyExists (p. 265)
- S3PrefixNotEmpty (p. 268)
- ShellCommandPrecondition (p. 271)

DynamoDBDataExists

A precondition to check that data exists in a DynamoDB table.

Syntax

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>
This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>The DynamoDB table to check.</td>
<td>String</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This object includes the following fields from the `Precondition` object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>The activity or data node for which this precondition is being checked. This is a runtime slot.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>preconditionTimeout</td>
<td>The precondition will be retried until the retryTimeout with a gap of retryDelay between attempts.</td>
<td>Time period; for example, &quot;1 hour&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>role</td>
<td>The IAM role to use for this precondition.</td>
<td>String</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This object includes the following fields from `RunnableObject`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>@activeInstances</td>
<td>Record of the currently scheduled instance objects</td>
<td>Schedulable object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptStatus</td>
<td>The status most recently reported from the object.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@actualEndTime</td>
<td>The date and time that the scheduled run actually ended. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@actualStartTime</td>
<td>The date and time that the scheduled run actually started. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptTimeout</td>
<td>The timeout time interval for an object attempt. If an attempt does not complete within the start time plus this time interval, AWS Data Pipeline marks the attempt as failed and your retry settings determine the next steps taken.</td>
<td>Time period; for example, &quot;1 hour&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@cascadeFailedOn</td>
<td>Description of which dependency the object failed on.</td>
<td>List of objects (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@componentParent</td>
<td>The component from which this instance is created.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorId</td>
<td>If the object failed, the error code. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorMessage</td>
<td>If the object failed, the error message. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorStackTrace</td>
<td>If the object failed, the error stack trace.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>failureAndRerunMode</td>
<td>Determines whether pipeline object failures and rerun commands cascade through pipeline object dependencies. For more information, see Cascading Failures and Reruns (p. 51).</td>
<td>String. Possible values are cascade and none.</td>
<td>No</td>
</tr>
<tr>
<td>@headAttempt</td>
<td>The latest attempt on the given instance.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@hostname</td>
<td>The host name of client that picked up the task attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>lateAfterTimeout</td>
<td>The time period in which the object run must start. If the object does not start within the scheduled start time plus this time interval, it is considered late.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>maximumRetries</td>
<td>The maximum number of times to retry the action. The default value is 2, which results in 3 tries total (1 original attempt plus 2 retries). The maximum value is 5 (6 total attempts).</td>
<td>Integer</td>
<td>No</td>
</tr>
<tr>
<td>onFail</td>
<td>An action to run when the current object fails.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onLateAction</td>
<td>The SnsAlarm to use when the object's run is late.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>An action to run when the current object succeeds.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>@reportProgressTime</td>
<td>The last time that Task Runner, or other code that is processing the tasks, called the ReportTaskProgress API.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>reportProgressTimeout</td>
<td>The time period for successive calls from Task Runner to the ReportTaskProgress API. If Task Runner, or other code that is processing the tasks, does not report progress within the specified period, the object attempt can be retried.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@resource</td>
<td>The resource instance on which the given activity/precondition attempt is being run.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>retryDelay</td>
<td>The timeout duration between two retry attempts. The default is 10 minutes.</td>
<td>Period. Minimum is &quot;1 second&quot;.</td>
<td>Yes</td>
</tr>
<tr>
<td>@scheduledEndTime</td>
<td>The date and time that the run was scheduled to end.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@scheduledStartTime</td>
<td>The date and time that the run was scheduled to start.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------</td>
<td>----------</td>
</tr>
<tr>
<td>@status</td>
<td>The status of this object. This is a runtime slot. Possible values are: pending, waiting_on_dependencies, running, waiting_on_runner, successful, and failed.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@triesLeft</td>
<td>The number of attempted runs remaining before setting the status of this object to failed. This is a runtime slot.</td>
<td>Integer (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@waitingOn</td>
<td>A list of all objects that this object is waiting on before it can enter the RUNNING state.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>

**DynamoDBTableExists**

A precondition to check that the DynamoDB table exists.

**Syntax**

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>The DynamoDB table to check.</td>
<td>String</td>
<td>Yes</td>
</tr>
</tbody>
</table>
This object includes the following fields from the **Precondition** object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>The activity or data node for which this precondition is being checked. This is a runtime slot.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>preconditionTimeout</td>
<td>The precondition will be retried until the retryTimeout with a gap of retryDelay between attempts.</td>
<td>Time period; for example, “1 hour”.</td>
<td>No</td>
</tr>
<tr>
<td>role</td>
<td>The IAM role to use for this precondition.</td>
<td>String</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This object includes the following fields from **RunnableObject**.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>@activeInstances</td>
<td>Record of the currently scheduled instance objects</td>
<td>Schedulable object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptStatus</td>
<td>The status most recently reported from the object.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@actualEndTime</td>
<td>The date and time that the scheduled run actually ended. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@actualStartTime</td>
<td>The date and time that the scheduled run actually started. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptTimeout</td>
<td>The timeout time interval for an object attempt. If an attempt does not complete within the start time plus this time interval, AWS Data Pipeline marks the attempt as failed and your retry settings determine the next steps taken.</td>
<td>Time period; for example, “1 hour”.</td>
<td>No</td>
</tr>
<tr>
<td>@cascadeFailedOn</td>
<td>Description of which dependency the object failed on.</td>
<td>List of objects (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@componentParent</td>
<td>The component from which this instance is created.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorId</td>
<td>If the object failed, the error code. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorMessage</td>
<td>If the object failed, the error message. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorStackTrace</td>
<td>If the object failed, the error stack trace.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>failureAndRerunMode</td>
<td>Determines whether pipeline object failures and rerun commands cascade through pipeline object dependencies. For more information, see Cascading Failures and Reruns (p. 51).</td>
<td>String. Possible values are cascade and none.</td>
<td>No</td>
</tr>
<tr>
<td>@headAttempt</td>
<td>The latest attempt on the given instance.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-----------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>@hostname</td>
<td>The host name of client that picked up the task attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>lateAfterTimeout</td>
<td>The time period in which the object run must start. If the object does not start within the scheduled start time plus this time interval, it is considered late.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>maximumRetries</td>
<td>The maximum number of times to retry the action. The default value is 2, which results in 3 tries total (1 original attempt plus 2 retries). The maximum value is 5 (6 total attempts).</td>
<td>Integer</td>
<td>No</td>
</tr>
<tr>
<td>onFail</td>
<td>An action to run when the current object fails.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onLateAction</td>
<td>The SnsAlarm to use when the object's run is late.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>An action to run when the current object succeeds.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>@reportProgressTime</td>
<td>The last time that Task Runner, or other code that is processing the tasks, called the ReportTaskProgress API.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>reportProgressTimeout</td>
<td>The time period for successive calls from Task Runner to the ReportTaskProgress API. If Task Runner, or other code that is processing the tasks, does not report progress within the specified period, the object attempt can be retried.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@resource</td>
<td>The resource instance on which the given activity/precondition attempt is being run.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>retryDelay</td>
<td>The timeout duration between two retry attempts. The default is 10 minutes.</td>
<td>Period. Minimum is &quot;1 second&quot;.</td>
<td>Yes</td>
</tr>
<tr>
<td>@scheduledEndTime</td>
<td>The date and time that the run was scheduled to end.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@scheduledStartTime</td>
<td>The date and time that the run was scheduled to start.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@status</td>
<td>The status of this object. This is a runtime slot. Possible values are: pending, waiting_on_dependencies, running, waiting_on_runner, successful, and failed.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@triesLeft</td>
<td>The number of attempted runs remaining before setting the status of this object to failed. This is a runtime slot.</td>
<td>Integer (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>@waitingOn</td>
<td>A list of all objects that this object is waiting on before it can enter the RUNNING state.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>

### Exists

Checks whether a data node object exists.

**Note**

We recommend that you use system-managed preconditions instead. For more information, see [Preconditions](#) (p. 15).

### 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.

```json
{
    "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

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-----------------------</td>
<td>----------</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields from the `Precondition` object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>The activity or data node for which this precondition is being checked. This is a runtime slot.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>preconditionTimeout</td>
<td>The precondition will be retried until the retryTimeout with a gap of retryDelay between attempts.</td>
<td>Time period; for example, &quot;1 hour&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>role</td>
<td>The IAM role to use for this precondition.</td>
<td>String</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This object includes the following fields from `RunnableObject`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>@activeInstances</td>
<td>Record of the currently scheduled instance objects</td>
<td>Schedulable object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptStatus</td>
<td>The status most recently reported from the object.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@actualEndTime</td>
<td>The date and time that the scheduled run actually ended. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@actualStartTime</td>
<td>The date and time that the scheduled run actually started. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptTimeout</td>
<td>The timeout time interval for an object attempt. If an attempt does not complete within the start time plus this time interval, AWS Data Pipeline marks the attempt as failed and your retry settings determine the next steps taken.</td>
<td>Time period; for example, &quot;1 hour&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@cascadeFailedOn</td>
<td>Description of which dependency the object failed on.</td>
<td>List of objects (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@componentParent</td>
<td>The component from which this instance is created.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorId</td>
<td>If the object failed, the error code. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorMessage</td>
<td>If the object failed, the error message. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorStackTrace</td>
<td>If the object failed, the error stack trace.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>failureAndRerunMode</td>
<td>Determines whether pipeline object failures and rerun commands cascade through pipeline object dependencies. For more information, see Cascading Failures and Reruns (p. 51).</td>
<td>String. Possible values are <code>cascade</code> and <code>none</code>.</td>
<td>No</td>
</tr>
<tr>
<td>@headAttempt</td>
<td>The latest attempt on the given instance.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@hostname</td>
<td>The host name of client that picked up the task attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>lateAfterTimeout</td>
<td>The time period in which the object run must start. If the object does not start within the scheduled start time plus this time interval, it is considered late.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>maximumRetries</td>
<td>The maximum number of times to retry the action. The default value is 2, which results in 3 tries total (1 original attempt plus 2 retries). The maximum value is 5 (6 total attempts).</td>
<td>Integer</td>
<td>No</td>
</tr>
<tr>
<td>onFail</td>
<td>An action to run when the current object fails.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onLateAction</td>
<td>The SnsAlarm to use when the object’s run is late.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>An action to run when the current object succeeds.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>@reportProgressTime</td>
<td>The last time that Task Runner, or other code that is processing the tasks, called the ReportTaskProgress API.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>reportProgressTimeout</td>
<td>The time period for successive calls from Task Runner to the ReportTaskProgress API. If Task Runner, or other code that is processing the tasks, does not report progress within the specified period, the object attempt can be retried.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@resource</td>
<td>The resource instance on which the given activity/precondition attempt is being run.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>retryDelay</td>
<td>The timeout duration between two retry attempts. The default is 10 minutes.</td>
<td>Period. Minimum is &quot;1 second&quot;.</td>
<td>Yes</td>
</tr>
<tr>
<td>@scheduledEndTime</td>
<td>The date and time that the run was scheduled to end.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@scheduledStartTime</td>
<td>The date and time that the run was scheduled to start.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>
### S3KeyExists

Checks whether a key exists in an Amazon S3 data node.

#### Syntax

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>
This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>s3Key</td>
<td>Amazon S3 key to check for existence.</td>
<td>String, for example “s3://examples-bucket/key”.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This object includes the following fields from the Precondition object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>The activity or data node for which this precondition is being checked. This is a runtime slot.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>preconditionTimeout</td>
<td>The precondition will be retried until the retryTimeout with a gap of retryDelay between attempts.</td>
<td>Time period; for example, “1 hour”.</td>
<td>No</td>
</tr>
<tr>
<td>role</td>
<td>The IAM role to use for this precondition.</td>
<td>String</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This object includes the following fields from RunnableObject.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>@activeInstances</td>
<td>Record of the currently scheduled instance objects</td>
<td>Schedulable object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptStatus</td>
<td>The status most recently reported from the object.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@actualEndTime</td>
<td>The date and time that the scheduled run actually ended. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@actualStartTime</td>
<td>The date and time that the scheduled run actually started. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptTimeout</td>
<td>The timeout time interval for an object attempt. If an attempt does not complete within the start time plus this time interval, AWS Data Pipeline marks the attempt as failed and your retry settings determine the next steps taken.</td>
<td>Time period; for example, “1 hour”.</td>
<td>No</td>
</tr>
<tr>
<td>@cascadeFailedOn</td>
<td>Description of which dependency the object failed on.</td>
<td>List of objects (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@componentParent</td>
<td>The component from which this instance is created.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorId</td>
<td>If the object failed, the error code. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorMessage</td>
<td>If the object failed, the error message. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorStackTrace</td>
<td>If the object failed, the error stack trace.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>failureAndRerunMode</td>
<td>Determines whether pipeline object failures and rerun commands cascade through pipeline object dependencies. For more information, see Cascading Failures and Reruns (p. 51).</td>
<td>String. Possible values are cascade and none.</td>
<td>No</td>
</tr>
<tr>
<td>@headAttempt</td>
<td>The latest attempt on the given instance.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@hostname</td>
<td>The host name of client that picked up the task attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>lateAfterTimeout</td>
<td>The time period in which the object run must start. If the object does not start within the scheduled start time plus this time interval, it is considered late.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>maximumRetries</td>
<td>The maximum number of times to retry the action. The default value is 2, which results in 3 tries total (1 original attempt plus 2 retries). The maximum value is 5 (6 total attempts).</td>
<td>Integer</td>
<td>No</td>
</tr>
<tr>
<td>onFail</td>
<td>An action to run when the current object fails.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onLateAction</td>
<td>The SnsAlarm to use when the object’s run is late.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>An action to run when the current object succeeds.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>@reportProgressTime</td>
<td>The last time that Task Runner, or other code that is processing the tasks, called the ReportTaskProgress API.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>reportProgressTimeout</td>
<td>The time period for successive calls from Task Runner to the ReportTaskProgress API. If Task Runner, or other code that is processing the tasks, does not report progress within the specified period, the object attempt can be retried.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@resource</td>
<td>The resource instance on which the given activity/precondition attempt is being run.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>retryDelay</td>
<td>The timeout duration between two retry attempts. The default is 10 minutes.</td>
<td>Period. Minimum is &quot;1 second&quot;.</td>
<td>Yes</td>
</tr>
<tr>
<td>@scheduledEndTime</td>
<td>The date and time that the run was scheduled to end.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@scheduledStartTime</td>
<td>The date and time that the run was scheduled to start.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>
See Also

• ShellCommandPrecondition (p. 271)

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.

```json
{
   "id" : "InputReady",
   "type" : "S3PrefixNotEmpty",
   "role" : "test-role",
   "s3Prefix" : "#{node.filePath}"
}
```

Syntax

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>s3Prefix</td>
<td>The Amazon S3 prefix to check for existence of objects.</td>
<td>String, for example &quot;s3://example-bucket/prefix&quot;</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This object includes the following fields from the Precondition object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>The activity or data node for which this precondition is being checked. This is a runtime slot.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>preconditionTimeout</td>
<td>The precondition will be retried until the retryTimeout with a gap of retryDelay between attempts.</td>
<td>Time period; for example, &quot;1 hour&quot;</td>
<td>No</td>
</tr>
<tr>
<td>role</td>
<td>The IAM role to use for this precondition.</td>
<td>String</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This object includes the following fields from RunnableObject.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>@activeInstances</td>
<td>Record of the currently scheduled instance objects</td>
<td>Schedulable object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptStatus</td>
<td>The status most recently reported from the object.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@actualEndTime</td>
<td>The date and time that the scheduled run actually ended. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@actualStartTime</td>
<td>The date and time that the scheduled run actually started. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptTimeout</td>
<td>The timeout time interval for an object attempt. If an attempt does not complete within the start time plus this time interval, AWS Data Pipeline marks the attempt as failed and your retry settings determine the next steps taken.</td>
<td>Time period; for example, &quot;1 hour&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>@cascadeFailedOn</td>
<td>Description of which dependency the object failed on.</td>
<td>List of objects (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@componentParent</td>
<td>The component from which this instance is created.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorId</td>
<td>If the object failed, the error code. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorMessage</td>
<td>If the object failed, the error message. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorStackTrace</td>
<td>If the object failed, the error stack trace.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>failureAndRerunMode</td>
<td>Determines whether pipeline object failures and rerun commands cascade through pipeline object dependencies. For more information, see Cascading Failures and Reruns (p. 51).</td>
<td>String. Possible values are <code>cascade</code> and <code>none</code>.</td>
<td>No</td>
</tr>
<tr>
<td>@headAttempt</td>
<td>The latest attempt on the given instance.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@hostname</td>
<td>The host name of client that picked up the task attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>lateAfterTimeout</td>
<td>The time period in which the object run must start. If the object does not start within the scheduled start time plus this time interval, it is considered late.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>onFail</td>
<td>An action to run when the current object fails.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onLateAction</td>
<td>The SnsAlarm to use when the object’s run is late.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>An action to run when the current object succeeds.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>@reportProgressTime</td>
<td>The last time that Task Runner, or other code that is processing the tasks, called the ReportTaskProgress API.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>reportProgressTimeout</td>
<td>The time period for successive calls from Task Runner to the ReportTaskProgress API. If Task Runner, or other code that is processing the tasks, does not report progress within the specified period, the object attempt can be retried.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@resource</td>
<td>The resource instance on which the given activity/precondition attempt is being run.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>retryDelay</td>
<td>The timeout duration between two retry attempts. The default is 10 minutes.</td>
<td>Period. Minimum is &quot;1 second&quot;.</td>
<td>Yes</td>
</tr>
</tbody>
</table>
**Name** | **Description** | **Type** | **Required**
--- | --- | --- | ---
@scheduledEndTime | The date and time that the run was scheduled to end. | DateTime (read-only) | No
@scheduledStartTime | The date and time that the run was scheduled to start. | DateTime (read-only) | No
@status | The status of this object. This is a runtime slot. Possible values are: pending, waiting_on_dependencies, running, waiting_on_runner, successful, and failed. | String (read-only) | No
@triesLeft | The number of attempted runs remaining before setting the status of this object to failed. This is a runtime slot. | Integer (read-only) | No
@waitingOn | A list of all objects that this object is waiting on before it can enter the RUNNING state. | String (read-only) | No

**See Also**
- ShellCommandPrecondition (p. 271)

**ShellCommandPrecondition**

A Unix/Linux shell command that can be run as a precondition.

**Example**

The following is an example of this object type.

```json
{
  "id" : "VerifyDataReadiness",
  "type" : "ShellCommandPrecondition",
  "command" : "perl check-data-ready.pl"
}
```

**Syntax**

The following fields are included in all objects.

**Name** | **Description** | **Type** | **Required**
--- | --- | --- | ---
id | The ID of the object. IDs must be unique within a pipeline definition. | String | Yes
name | The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id. | String | No
parent | The parent of the object. | Object reference | No
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>command</td>
<td>The command to run. This value and any associated parameters must function in the environment from which you are running the Task Runner.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>scriptArgument</td>
<td>A list of arguments to pass to the shell script.</td>
<td>List of strings</td>
<td>No</td>
</tr>
<tr>
<td>scriptUri</td>
<td>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.</td>
<td>A valid S3 URI</td>
<td>No</td>
</tr>
<tr>
<td>stderr</td>
<td>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.</td>
<td>String. For example, 's3://example-stdout'</td>
<td>No</td>
</tr>
<tr>
<td>stdout</td>
<td>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.</td>
<td>String. For example, 's3://example-stdout'</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields from the Precondition object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>The activity or data node for which this precondition is being checked. This is a runtime slot.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>preconditionTimeout</td>
<td>The precondition will be retried until the retryTimeout with a gap of retryDelay between attempts.</td>
<td>Time period; for example, &quot;1 hour&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>role</td>
<td>The IAM role to use for this precondition.</td>
<td>String</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This object includes the following fields from RunnableObject.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>@activeInstances</td>
<td>Record of the currently scheduled instance objects</td>
<td>Schedulable object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptStatus</td>
<td>The status most recently reported from the object.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@actualEndTime</td>
<td>The date and time that the scheduled run actually ended. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@actualStartTime</td>
<td>The date and time that the scheduled run actually started. This is a runtime slot.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>attemptTimeout</td>
<td>The timeout time interval for an object attempt. If an attempt does not complete within the start time plus this time interval, AWS Data Pipeline marks the attempt as failed and your retry settings determine the next steps taken.</td>
<td>Time period; for example, &quot;1 hour&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@cascadeFailedOn</td>
<td>Description of which dependency the object failed on.</td>
<td>List of objects (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@componentParent</td>
<td>The component from which this instance is created.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorId</td>
<td>If the object failed, the error code. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorMessage</td>
<td>If the object failed, the error message. This is a runtime slot.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>errorStackTrace</td>
<td>If the object failed, the error stack trace.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>failureAndRerunMode</td>
<td>Determines whether pipeline object failures and rerun commands cascade through pipeline object dependencies. For more information, see Cascading Failures and Reruns (p. 51).</td>
<td>String. Possible values are cascade and none.</td>
<td>No</td>
</tr>
<tr>
<td>@headAttempt</td>
<td>The latest attempt on the given instance.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@hostname</td>
<td>The host name of client that picked up the task attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>lateAfterTimeout</td>
<td>The time period in which the object run must start. If the object does not start within the scheduled start time plus this time interval, it is considered late.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>maximumRetries</td>
<td>The maximum number of times to retry the action. The default value is 2, which results in 3 tries total (1 original attempt plus 2 retries). The maximum value is 5 (6 total attempts).</td>
<td>Integer</td>
<td>No</td>
</tr>
<tr>
<td>onFail</td>
<td>An action to run when the current object fails.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onLateAction</td>
<td>The SnsAlarm to use when the object's run is late.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>onSuccess</td>
<td>An action to run when the current object succeeds.</td>
<td>List of SnsAlarm (p. 289) object references</td>
<td>No</td>
</tr>
<tr>
<td>@reportProgressTime</td>
<td>The last time that Task Runner, or other code that is processing the tasks, called the ReportTaskProgress API.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>reportProgressTimeout</td>
<td>The time period for successive calls from Task Runner to the ReportTaskProgress API. If Task Runner, or other code that is processing the tasks, does not report progress within the specified period, the object attempt can be retried.</td>
<td>Time period; for example, &quot;1 hour&quot;. The minimum value is &quot;15 minutes&quot;.</td>
<td>No</td>
</tr>
<tr>
<td>@resource</td>
<td>The resource instance on which the given activity/precondition attempt is being run.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>retryDelay</td>
<td>The timeout duration between two retry attempts. The default is 10 minutes.</td>
<td>Period. Minimum is &quot;1 second&quot;.</td>
<td>Yes</td>
</tr>
<tr>
<td>@scheduledEndTime</td>
<td>The date and time that the run was scheduled to end.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@scheduledStartTime</td>
<td>The date and time that the run was scheduled to start.</td>
<td>DateTime (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@status</td>
<td>The status of this object. This is a runtime slot. Possible values are: pending, waiting_on_dependencies, running, waiting_on_runner, successful, and failed.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@triesLeft</td>
<td>The number of attempted runs remaining before setting the status of this object to failed. This is a runtime slot.</td>
<td>Integer (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>@waitingOn</td>
<td>A list of all objects that this object is waiting on before it can enter the RUNNING state.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>
See Also

- ShellCommandActivity (p. 233)
- Exists (p. 262)

Databases

The following are Data Pipeline Databases:

Topics

- JdbcDatabase (p. 275)
- RdsDatabase (p. 276)
- RedshiftDatabase (p. 277)

JdbcDatabase

Defines a JDBC database.

Example

The following is an example of this object type.

```json
{
  "id" : "MyJdbcDatabase",
  "type" : "JdbcDatabase",
  "connectionString" : "jdbc:mysql://hostname:portname/dbname",
  "username" : "user_name",
  "*password" : "my_password"
}
```

Syntax

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------------------------------------------------</td>
<td>-----------</td>
<td>----------</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>connectionString</td>
<td>The JDBC connection string to access the database.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>jdbcDriverClass</td>
<td>The driver class to load before establishing the JDBC connection.</td>
<td>String</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This object includes the following fields from the `Database` object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>databaseName</td>
<td>The name of the logical database.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>jdbcProperties</td>
<td>The properties of the JDBC connections for this database.</td>
<td>List of strings</td>
<td>No</td>
</tr>
<tr>
<td>*password</td>
<td>The password to connect to the database.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>username</td>
<td>The user name to connect to the database.</td>
<td>String</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**RdsDatabase**

Defines an Amazon RDS database.

**Note**

RdsDatabase can only be associated with a MySqlDataNode.

**Example**

The following is an example of this object type.

```json
{
    "id" : "MyRdsDatabase",
    "type" : "RdsDatabase",
    "username" : "user_name",
    "*password" : "my_password",
    "databaseName" : "database_name"
}
```
Syntax

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields from the Database object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>databaseName</td>
<td>The name of the logical database.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>jdbcProperties</td>
<td>The properties of the JDBC connections for this database.</td>
<td>List of strings</td>
<td>No</td>
</tr>
<tr>
<td>*password</td>
<td>The password to connect to the database.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>username</td>
<td>The user name to connect to the database.</td>
<td>String</td>
<td>Yes</td>
</tr>
</tbody>
</table>

RedshiftDatabase

Defines an Amazon Redshift database.

Example

The following is an example of this object type.

```json
{
    "id" : "MyRedshiftDatabase",
    "type" : "RedshiftDatabase",
    "clusterId" : "clusterId",
    "username" : "user_name",
    "*password" : "my_password",
}
"databaseName" : "database_name"

**Syntax**

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>clusterId</td>
<td>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 clusterId value is mydb. In the Amazon Redshift console, this value is &quot;Cluster Name&quot;.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>connectionString</td>
<td>The JDBC endpoint for connecting to an Amazon Redshift instance owned by an account different than the pipeline.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields from the **Database** object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>databaseName</td>
<td>The name of the logical database.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>
Data Formats

The following are Data Pipeline Data Formats:

Topics
- CSV Data Format (p. 279)
- Custom Data Format (p. 280)
- DynamoDBDataFormat (p. 282)
- DynamoDBExportDataFormat (p. 284)
- RegEx Data Format (p. 286)
- TSV Data Format (p. 288)

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.

```json
{
    "id" : "MyOutputDataType",
    "type" : "CSV",
    "column" : [
        "Name STRING",
        "Score INT",
        "DateOfBirth TIMESTAMP"
    ]
}
```

Syntax

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### 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.

```json
{
  "id" : "MyOutputDataType",
  "type" : "Custom",
  "columnSeparator" : "",
  "column" : [
    { "Name STRING", "Score INT", "DateOfBirth TIMESTAMP" }
  ]
}
```
"recordSeparator" : "\n",
"column" : [
  "Name STRING",
  "Score INT",
  "DateOfBirth TIMESTAMP"
]
}

Syntax

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>The structure of the data file. Use column names and data types separated by a space. For example:</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>[ &quot;Name STRING&quot;, &quot;Score INT&quot;, &quot;DateOfBirth TIMESTAMP&quot; ]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>You can omit the data type when using STRING, which is the default.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Valid data types: TINYINT, SMALLINT, INT, BIGINT, BOOLEAN, FLOAT, DOUBLE, STRING, TIMESTAMP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>columnSeparator</td>
<td>A character that indicates the end of a column in a data file, for example &quot;,&quot;.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Type</td>
<td>Required</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>recordSeparator</td>
<td>A character that indicates the end of a row in a data file, for example &quot;\n&quot;. Multi-byte Unicode characters are possible using the example &quot;\u0002\n&quot; where the Unicode character number 0002 combines with a newline character.</td>
<td>String</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**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. 284).

**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.

```json
{
   "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**

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>
This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>The structure of the data file. Use column names and data types separated by a space. For example:</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>[ &quot;Name STRING&quot;, &quot;Score BIGINT&quot;, &quot;Ratio DOUBLE&quot; ]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Valid data types: TINYINT, SMALLINT, BIGINT, BOOLEAN, FLOAT, DOUBLE, STRING</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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

**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.

```json
{
    "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",
```

API Version 2012-10-29

284
"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**

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>
### RegEx Data Format

A custom data format defined by a regular expression.

**Example**

The following is an example of this object type.

```json
{
    "id": "MyInputDataType",
    "type": "RegEx",
    "inputRegEx": "([^ ]+) (\S+) (\S+) (\S+) (\S+) (\S+) (\S+) (\S+) (\S+)
    (-|\[0-9\]) (-|\[0-9\]) (?: (\S+) (\S+) (\S+) (\S+) (\S+) (\S+) (\S+) (\S+) (\S+))?",
    "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

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>The structure of the data file. Use column names and data types separated by a space. For example: [ &quot;Name STRING&quot;, &quot;Score INT&quot;, &quot;DateOfBirth TIMESTAMP&quot; ] You can omit the data type when using STRING, which is the default. Valid data types: TINYINT, SMALLINT, INT, BIGINT, BOOLEAN, FLOAT, DOUBLE, STRING, TIMESTAMP</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>inputRegEx</td>
<td>The regular expression to parse an S3 input file. inputRegEx provides a way to retrieve columns from relatively unstructured data in a file.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>outputFormat</td>
<td>The column fields retrieved by inputRegEx, but referenced as %1, %2, %3, etc. using Java formatter syntax. For more information, see Format String Syntax</td>
<td>String</td>
<td>Yes</td>
</tr>
</tbody>
</table>
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.

```json
{
  "id" : "MyOutputDataType",
  "type" : "TSV",
  "column" : [
    "Name STRING",
    "Score INT",
    "DateOfBirth TIMESTAMP"
  ]
}
```

Syntax

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>
This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>The structure of the data file. Use column names and data types separated by a space. For example: [ &quot;Name STRING&quot;, &quot;Score INT&quot;, &quot;DateOfBirth TIMESTAMP&quot; ] You can omit the data type when using STRING, which is the default. Valid data types: TINYINT, SMALLINT, INT, BIGINT, BOOLEAN, FLOAT, DOUBLE, STRING, TIMESTAMP</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>escapeChar</td>
<td>A character, for example &quot;,&quot;, that instructs the parser to ignore the next character.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

**Actions**

The following are Data Pipeline Actions:

**Topics**

- SnsAlarm (p. 289)
- Terminate (p. 291)

**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.

```json
{
    "id" : "SuccessNotify",
    "type" : "SnsAlarm",
    "subject" : "COPY SUCCESS: #{node.@scheduledStartTime}"
    "message" : "Files were copied from #{node.input} to #{node.output}."
}
```
Syntax

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>for an object in a pipeline definition, AWS Data Pipeline automatically</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>duplicates the value of id.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>as Pipeline, Component, Instance, or Attempt.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>

This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>The body text of the Amazon SNS notification.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>role</td>
<td>The IAM role to use to create the Amazon SNS alarm.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>subject</td>
<td>The subject line of the Amazon SNS notification message.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>topicArn</td>
<td>The destination Amazon SNS topic ARN for the message.</td>
<td>String</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This object includes the following fields from the Action object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>The node for which this action is being performed. This is a runtime slot.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>
**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 finish 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 how the activity is considered late.

```json
{
   "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**

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>
RequiredTypeDescriptionName

@sphere The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt. String (read-only) No

type The type of object. Use one of the predefined AWS Data Pipeline object types. String Yes

version Pipeline version the object was created with. String No

This object includes the following fields from the Action object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>The node for which this action is being performed. This is a runtime slot.</td>
<td>Object reference (read-only)</td>
<td>No</td>
</tr>
</tbody>
</table>

**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. To override this behavior, use the --force parameter from the CLI.

**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. For more information about specifying start and end times, see Time Zones (p. 21).

```json
{
  "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.

```json
{
  "id": "SchedulePeriod",
  "name": "SchedulePeriod",
  "startAt": "FIRST_ACTIVATION_DATE_TIME",
  "period": "1 hours",
  "type": "Schedule",
}```
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"
}
```

**Syntax**

The following fields are included in all objects.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>The ID of the object. IDs must be unique within a pipeline definition.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>name</td>
<td>The optional, user-defined label of the object. If you do not provide a name for an object in a pipeline definition, AWS Data Pipeline automatically duplicates the value of id.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>parent</td>
<td>The parent of the object.</td>
<td>Object reference</td>
<td>No</td>
</tr>
<tr>
<td>pipelineId</td>
<td>The ID of the pipeline to which this object belongs.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>@sphere</td>
<td>The sphere of an object denotes its place in the pipeline lifecycle, such as Pipeline, Component, Instance, or Attempt.</td>
<td>String (read-only)</td>
<td>No</td>
</tr>
<tr>
<td>type</td>
<td>The type of object. Use one of the predefined AWS Data Pipeline object types.</td>
<td>String</td>
<td>Yes</td>
</tr>
<tr>
<td>version</td>
<td>Pipeline version the object was created with.</td>
<td>String</td>
<td>No</td>
</tr>
</tbody>
</table>
This object includes the following fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>endDateTime</td>
<td>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.</td>
<td>String</td>
<td>No</td>
</tr>
<tr>
<td>period</td>
<td>How often the pipeline should run. The format is &quot;N [minutes</td>
<td>hours</td>
<td>days</td>
</tr>
<tr>
<td>startDateTime</td>
<td>The date and time to start the scheduled runs. You must use either startDateTime or startAt but not both.</td>
<td>String</td>
<td>Yes (or use startAt)</td>
</tr>
<tr>
<td>startAt</td>
<td>The date and time at which to start the scheduled pipeline runs. Valid value is FIRST_ACTIVATION_DATE_TIME. FIRST_ACTIVATION_DATE_TIME is assumed to be the current date and time.</td>
<td>String</td>
<td>Yes (or use startDateTime)</td>
</tr>
<tr>
<td>occurrences</td>
<td>The number of times to execute the pipeline after it's activated. You can't use occurrences with endDateTime.</td>
<td>Integer</td>
<td>No</td>
</tr>
</tbody>
</table>
AWS Data Pipeline CLI Reference

This is the reference for the AWS Data Pipeline command line interface (CLI).

**Important**
This CLI is deprecated and we will remove support for it in the future. Instead, you can use the AWS Command Line Interface (AWS CLI). For more information, see [http://aws.amazon.com/cli/](http://aws.amazon.com/cli/).

**Contents**
- Install the CLI (p. 295)
- Command Line Syntax (p. 300)
- --activate (p. 300)
- --cancel (p. 302)
- --create (p. 303)
- --delete (p. 304)
- --get, --g (p. 305)
- --help, --h (p. 306)
- --list-pipelines (p. 307)
- --list-runs (p. 307)
- --mark-finished (p. 309)
- --put (p. 310)
- --rerun (p. 311)
- --validate (p. 312)
- Common Options (p. 313)
- Creating a Pipeline (p. 314)
- Example Pipeline Definition Files (p. 316)

**Install the AWS Data Pipeline Command Line Interface**

You can use the AWS Data Pipeline command line interface (CLI) to create and manage pipelines. This CLI is written in Ruby and makes calls to the web service on your behalf.
Important
AWS Data Pipeline is now supported through the AWS Command Line Interface. The Ruby-based client is deprecated and support will be removed in a future release. To install the AWS Command Line Interface see, http://aws.amazon.com/cli/.

To install the CLI, complete the following tasks:

Tasks
- Install Ruby (p. 296)
- Install RubyGems (p. 297)
- Install the Required Ruby Gems (p. 297)
- Install the CLI (p. 298)
- Configure Credentials for the CLI (p. 298)

Install Ruby

The AWS Data Pipeline CLI requires Ruby 1.9.3 or later. Some operating systems come with Ruby pre-installed, but it might be an earlier version number.

To check whether Ruby is installed, run the following command. If Ruby is installed, this command displays the version information.

```
ruby -v
```

Linux/Unix/Mac OS

If you don't have Ruby 1.9.3 or later installed on Linux/Unix/Mac OS, download and install it from http://www.ruby-lang.org/en/downloads/, and then follow the instructions for your operating system.

Windows

If you don't have Ruby 1.9.3 or later installed on Windows, use RubyInstaller to install Ruby on Windows. Download the RubyInstaller for Ruby 1.9.3 from http://rubyinstaller.org/downloads/, and then run the executable file. Be sure to select Add Ruby executables to your PATH.
Install RubyGems

The AWS Data Pipeline CLI requires RubyGems version 1.8 or later.

To check whether RubyGems is installed, run the following command. If RubyGems is installed, this command displays the version information.

```
gem -v
```

Linux/Unix/Mac OS

If you don't have RubyGems 1.8 or later installed on Linux/Unix/Mac OS, download and extract it from http://rubyforge.org/frs/?group_id=126. Navigate to the folder where you extracted RubyGems, and then install it using the following command.

```
sudo ruby setup.rb
```

Windows

If you don't have RubyGems 1.8 or later installed on Windows, install the Ruby Development Kit. You must download and extract the version of the Ruby DevKit that matches your version of Ruby from http://rubyinstaller.org/downloads/. For example, Ruby 1.9.3 requires version tdm-32-4.5.2. Navigate to the folder where you extracted the Ruby DevKit, and then install it using the following commands.

```
ruby dk.rb init
ruby dk.rb install
```

Install the Required Ruby Gems

The AWS Data Pipeline CLI requires the following Ruby gems:

- json
- uuidtools
- httparty
- bigdecimal
- nokogiri

For each gem, repeat the following process until all gems are installed. In each example command, replace `gem` with the name of the gem.

To check whether a gem is installed, run the following command. If the gem is installed, the command displays the name and version of the gem.

```
gem search gem
```

If you don't have the gem installed, then you must install it.

Linux/Unix/Mac OS

Install the gem using the following command.
Install the gem using the following command.

```
sudo gem install gem
```

Install the gem using the following command.

```
gem install gem
```

Install the CLI

After you have installed your Ruby environment, you're ready to install the AWS Data Pipeline CLI.

To install the AWS Data Pipeline CLI

2. Unzip the compressed file. For example, on Linux/Unix/Mac OS use the following command:

```
unzip datapipeline-cli.zip
```

This command uncompresses the CLI and supporting code into a new directory named `data-pipeline-cli`.

3. (Optional) If you add the `datapipeline-cli` directory to your `PATH`, you can use the CLI without specifying the complete path. In the examples in this reference, we assume that you've updated your `PATH`, or that you run the CLI from the directory where it's installed.

Configure Credentials for the CLI

To connect to the AWS Data Pipeline web service to process your commands, the CLI needs the credentials of an AWS account that has permissions to create or manage pipelines.

For CLI access, you need an access key ID and secret access key. Use IAM user access keys instead of AWS root account access keys. IAM lets you securely control access to AWS services and resources in your AWS account. For more information about creating access keys, see How Do I Get Security Credentials? in the AWS General Reference.

You can pass credentials to the CLI using one of the following methods:

- Implicitly, specifying a JSON file in a known location
- Explicitly, specifying a JSON file on the command line
- Explicitly, specifying multiple command-line options

To create a credentials file

A credentials file contains the following name-value pairs:

```
comment
  An optional comment.
access-id
  The access key ID.
private-key
  The secret access key.
```
endpoint
The endpoint for AWS Data Pipeline for the region to use.

log-uri
The location of the Amazon S3 bucket where AWS Data Pipeline writes log files.

The following is an example JSON file for the us-east-1 region. Replace access_key_id and secret_access_key with the appropriate credentials.

```
{
    "access-id": "access_key_id",
    "private-key": "secret_access_key",
    "endpoint": "https://datapipeline.us-east-1.amazonaws.com",
    "region": "us-east-1",
    "log-uri": "s3://myawsbucket/logfiles"
}
```

To pass credentials implicitly using a JSON file

This method is often the most convenient one. Create a JSON file named credentials.json in either your home directory, or the directory where CLI is installed. The CLI loads the credentials implicitly, and you need not specify credentials on the command line.

After setting up your credentials file, test the CLI using the following command. The command displays the list of pipelines that those credentials have been granted permission to access in the specified region.

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

If the CLI is installed and configured correctly, the command displays the following output.

```
Total of 0 pipelines.
```

To pass credentials explicitly using a JSON file

Create a JSON file named credentials.json and add the --credentials option to each call to specify the location of the JSON file. This method is useful if you are connecting to a machine using SSH to run the CLI remotely, or if you are testing different sets of credentials.

The following example command explicitly uses the credentials stored in the specified JSON file. The command displays the list of pipelines that those credentials have been granted permission to access in the specified region.

```
 datapipeline --list-pipelines --credentials /my-directory/credentials.json
```

If the CLI is installed and configured correctly, the command displays the following output.

```
Total of 0 pipelines.
```

To pass credentials using command-line options

Add the --access-key, --secret-key, and --endpoint options to each call to specify the credentials. Because you are passing credentials on the command line for every call, you should take additional precautions to ensure the privacy of your calls, such as clearing auto-complete when you are done with your terminal session and storing any scripts in a secure location.
The following example command explicitly uses credentials specified at the command line. Replace `my-access-key-id` with the access key ID, `my-secret-access-key` with the secret access key, and `endpoint` with the endpoint for the region to use. The command displays the list of pipelines that those credentials have been granted permission to access.

```
datapipeline --list-pipelines --access-key my-access-key-id --secret-key my-secret-access-key --endpoint endpoint
```

If the CLI is installed and configured correctly, the command displays the following output.

```
Total of 0 pipelines.
```

## Command Line Syntax

The syntax that you use to run the command line interface (CLI) differs slightly depending on the operating system that you use. Note that the examples on this page assume that you are running the commands from the directory where you unzipped the CLI.

In the Linux/Unix/Mac OS X version of the CLI, you use a period and slash (`./`) to indicate that the script is located in the current directory. The operating system automatically detects that the script is a Ruby script and uses the correct libraries to interpret the script. The following example shows you how to issue a command on Linux, Unix, and Mac OS X.

```
./datapipeline command [options]
```

In the Windows version of the CLI, using the current directory is implied, but you must explicitly specify the scripting engine to use with `ruby`. The following example shows you how to issue a command on Windows.

```
ruby datapipeline command [options]
```

For brevity, we use a simplified syntax in this documentation, rather than operating system-specific syntax. For example, the following is the simplified syntax for the command to display help for the CLI.

```
datapipeline --help
```

You can combine commands on a single command line. Commands are processed from left to right.

AWS Data Pipeline also supports a variety of complex expressions and functions within pipeline definitions. For more information, see [Pipeline Expressions and Functions](p. 161).

### --activate

**Description**

Starts a new or existing pipeline.
Syntax

datapipline --activate --force --id pipeline_id [Common Options]

Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--force</td>
<td>Override AWS Data Pipeline's attempt to prevent accidental backfills, which normally blocks pipeline activation if a pipeline component's scheduledStartTime is earlier than 1 day ago. Required: No</td>
</tr>
<tr>
<td>--id pipeline_id</td>
<td>The identifier of the pipeline.&lt;br&gt;You must specify the identifier of the pipeline when updating an existing pipeline with a new pipeline definition file. Required: Yes</td>
</tr>
<tr>
<td></td>
<td>Example: --id df-00627471SOVYZEXAMPLE</td>
</tr>
</tbody>
</table>

Common Options

For more information, see Common Options for AWS Data Pipeline Commands (p. 313).

Output

A response indicating that the new definition was successfully loaded, or, in the case where you are also using the --create (p. 303) command, an indication that the new pipeline was successfully created.

Examples

The following example shows how to use --activate to create a new pipeline, put the pipeline definition, and activate it:

datapipline --create my-pipeline --put my-pipeline-definition.json
Pipeline with name 'my-pipeline' and id 'df-00627471SOVYZEXAMPLE' created.<br>Pipeline definition 'my-pipeline-definition.json' uploaded.

Pipeline activated.

Related Commands

- --create (p. 303)
- --get, --g (p. 305)
--cancel

Description

Cancels one or more specified objects from within a pipeline that is either currently running or ran previously.

To see the status of the canceled pipeline object, use --list-runs.

Syntax

```
datapipeline --cancel object_id --id pipeline_id [Common Options]
```

Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object_id</td>
<td>The identifier of the object to cancel. You can specify the name of a single object, or a comma-separated list of object identifiers. Required: Yes Example: o-06198791C436IEXAMPLE</td>
</tr>
<tr>
<td>--id pipeline_id</td>
<td>The identifier of the pipeline. Required: Yes Example: --id df-00627471SOVYZEXAMPLE</td>
</tr>
</tbody>
</table>

Common Options

For more information, see Common Options for AWS Data Pipeline Commands (p. 313).

Output

None.

Examples

The following example demonstrates how to list the objects of a previously run or currently running pipeline. Next, the example cancels an object of the pipeline. Finally, the example lists the results of the canceled object.

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

datapipeline --id df-00627471SOVYZEXAMPLE --cancel o-06198791C436IEXAMPLE

datapipeline --list-runs --id df-00627471SOVYZEXAMPLE
```
Related Commands

- --delete (p. 304)
- --list-pipelines (p. 307)
- --list-runs (p. 307)

--create

Description

Creates a data pipeline with the specified name, but does not activate the pipeline.

There is a limit of 100 pipelines per AWS account.

To specify a pipeline definition file when you create the pipeline, use this command with the --put (p. 310) command.

Syntax

datapipeline --create name [Common Options]

Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the pipeline. Required: Yes Example: my-pipeline</td>
</tr>
</tbody>
</table>

Common Options

For more information, see Common Options for AWS Data Pipeline Commands (p. 313).

Output

Pipeline with name 'name' and id 'df-xxxxxxxxxxxxxxxxxxxx' created.

The identifier of the newly created pipeline (df-xxxxxxxxxxxxxxxxxxxx). You must specify this identifier with the --id command whenever you issue a command that operates on the corresponding pipeline.

Examples

The following example creates the first pipeline without specifying a pipeline definition file, and creates the second pipeline with a pipeline definition file.
Related Commands

- --delete (p. 304)
- --list-pipelines (p. 307)
- --put (p. 310)

--delete

Description

Stops the specified data pipeline, and cancels its future runs.

This command removes the pipeline definition file and run history. This action is irreversible; you can't restart a deleted pipeline.

Syntax

datapipeline --delete --id pipeline_id [Common Options]

Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--id pipeline_id</td>
<td>The identifier of the pipeline.</td>
</tr>
<tr>
<td></td>
<td>Required: Yes</td>
</tr>
<tr>
<td>Example: --id df-00627471SOVYZEXAMPLE</td>
<td></td>
</tr>
</tbody>
</table>

Common Options

For more information, see Common Options for AWS Data Pipeline Commands (p. 313).

Output

State of pipeline id 'df-xxxxxxxxxxxxxxxxxxxxx' is currently 'state'
Deleted pipeline 'df-xxxxxxxxxxxxxxxxxxxxx'

A message indicating that the pipeline was successfully deleted.
Examples

The following example deletes the pipeline with the identifier df-00627471SOVYZEXAMPLE.

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

Related Commands

- `--create (p. 303)`
- `--list-pipelines (p. 307)`

**--get, --g**

Description

Gets the pipeline definition file for the specified data pipeline and saves it to a file. If no file is specified, the file contents are written to standard output.

Syntax

```
datapipeline --get pipeline_definition_file --id pipeline_id --version pipeline_version [Common Options]
```

Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--id pipeline_id</code></td>
<td>The identifier of the pipeline. Required: Yes</td>
</tr>
<tr>
<td></td>
<td>Example: <code>--id df-00627471SOVYZEXAMPLE</code></td>
</tr>
<tr>
<td><code>pipeline_definition_file</code></td>
<td>The full path to the output file that receives the pipeline definition. Default: standard output</td>
</tr>
<tr>
<td></td>
<td>Required: No</td>
</tr>
<tr>
<td></td>
<td>Example: <code>my-pipeline.json</code></td>
</tr>
<tr>
<td><code>--version pipeline_version</code></td>
<td>The pipeline version. Required: No</td>
</tr>
<tr>
<td></td>
<td>Example: <code>--version active</code></td>
</tr>
<tr>
<td></td>
<td>Example: <code>--version latest</code></td>
</tr>
</tbody>
</table>
Common Options

For more information, see Common Options for AWS Data Pipeline Commands (p. 313).

Output

If an output file is specified, the output is a pipeline definition file; otherwise, the contents of the pipeline definition are written to standard output.

Examples

The first two command writes the definition to standard output and the second command writes the pipeline definition to the file my-pipeline.json.

```
 datapipeline --get --id df-00627471SOVYZEXAMPLE
 datapipeline --get my-pipeline.json --id df-00627471SOVYZEXAMPLE
```

Related Commands

• --create (p. 303)
• --put (p. 310)

--help, --h

Description

Displays information about the commands provided by the CLI.

Syntax

```
 datapipeline --help
```

Options

None.

Output

A list of the commands used by the CLI, printed to standard output.
## --list-pipelines

### Description
Lists the pipelines that you have permission to access.

### Syntax
```
datapipeline --list-pipelines
```

### Output
This command produces a list of pipelines created by the current user, including the name of the pipeline, the pipeline identifier, the state of the pipeline, and the user ID of the account that created the pipelines. For example:

<table>
<thead>
<tr>
<th>Name</th>
<th>Id</th>
<th>State</th>
<th>UserId</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyPipeline</td>
<td>df-00627471SOVYZEXAMPLE</td>
<td>PENDING</td>
<td>601204199999</td>
</tr>
</tbody>
</table>

### Options
None.

### Related Commands
- `--create` (p. 303)
- `--list-runs` (p. 307)

## --list-runs

### Description
Lists the times the specified pipeline has run. You can optionally filter the complete list of results to include only the runs you are interested in.

### Syntax
```
datapipeline --list-runs --id pipeline_id [filter] [Common Options]
```
Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>--id pipeline_id</strong></td>
<td>The identifier of the pipeline.</td>
</tr>
<tr>
<td></td>
<td>Required: Yes</td>
</tr>
<tr>
<td><strong>--status code</strong></td>
<td>Filters the list to include only runs in the specified statuses.</td>
</tr>
<tr>
<td></td>
<td>The valid statuses are as follows: waiting, pending, cancelled, running,</td>
</tr>
<tr>
<td></td>
<td>finished, failed, waiting_for_runner and waiting_on_dependencies.</td>
</tr>
<tr>
<td></td>
<td>Required: No</td>
</tr>
<tr>
<td></td>
<td>Example: <strong>--status running</strong></td>
</tr>
<tr>
<td></td>
<td>You can combine statuses as a comma-separated list.</td>
</tr>
<tr>
<td></td>
<td>Example: <strong>--status pending,waiting_on_dependencies</strong></td>
</tr>
<tr>
<td><strong>--failed</strong></td>
<td>Filters the list to include only runs in the failed state that started</td>
</tr>
<tr>
<td></td>
<td>during the last 2 days and were scheduled to end within the last 15 days.</td>
</tr>
<tr>
<td></td>
<td>Required: No</td>
</tr>
<tr>
<td><strong>--running</strong></td>
<td>Filters the list to include only runs in the running state that started</td>
</tr>
<tr>
<td></td>
<td>during the last 2 days and were scheduled to end within the last 15 days.</td>
</tr>
<tr>
<td></td>
<td>Required: No</td>
</tr>
<tr>
<td><strong>--start-interval</strong></td>
<td>Filters the list to include only runs that started within the specified</td>
</tr>
<tr>
<td>date1, date2</td>
<td>interval.</td>
</tr>
<tr>
<td></td>
<td>Required: No</td>
</tr>
<tr>
<td><strong>--schedule-interval</strong></td>
<td>Filters the list to include only runs that are scheduled to start within</td>
</tr>
<tr>
<td>date1, date2</td>
<td>the specified interval.</td>
</tr>
<tr>
<td></td>
<td>Required: No</td>
</tr>
</tbody>
</table>

Common Options

For more information, see Common Options for AWS Data Pipeline Commands (p. 313).

Output

A list of the times the specified pipeline has run and the status of each run. You can filter this list by the options you specify when you run the command.
**Examples**

The first command lists all the runs for the specified pipeline. The other commands show how to filter the complete list of runs using different options.

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

datapipeline --list-runs --id df-00627471SOVYZEXAMPLE --status PENDING

datapipeline --list-runs --id df-00627471SOVYZEXAMPLE --start-interval 2011-11-29T06:07:21,2011-12-06T06:07:21

datapipeline --list-runs --id df-00627471SOVYZEXAMPLE --schedule-interval 2011-11-29T06:07:21,2011-12-06T06:07:21

datapipeline --list-runs --id df-00627471SOVYZEXAMPLE --failed

datapipeline --list-runs --id df-00627471SOVYZEXAMPLE --running
```

**Related Commands**

- `--list-pipelines` (p. 307)

---

**--mark-finished**

**Description**

Marks one or more pipeline objects with the FINISHED status.

**Syntax**

```
datapipeline --mark-finished object_id [Common Options]
```

**Note**

`object_id` can be a comma-separated list.

**Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>object_id</code></td>
<td>The identifier of the object or comma-separated list of identifiers for multiple objects.</td>
</tr>
</tbody>
</table>

Required: Yes

Example: o-06198791C436IEXAMPLE
The identifier of the pipeline.

Required: Yes

Example: `--id df-00627471SOVYZEXAMPLE`

## Common Options

For more information, see Common Options for AWS Data Pipeline Commands (p. 313).

### --put

**Description**

Uploads a pipeline definition file to AWS Data Pipeline for a new or existing pipeline, but does not activate the pipeline. Use the `--activate` parameter in a separate command when you want the pipeline to begin.

To specify a pipeline definition file at the time that you create the pipeline, use this command with the `--create` command.

**Syntax**

```
datapipeline --put pipeline_definition_file --id pipeline_id [Common Options]
```

### Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--id pipeline_id</code></td>
<td>The identifier of the pipeline.</td>
</tr>
<tr>
<td></td>
<td>Required: Conditional</td>
</tr>
<tr>
<td></td>
<td>Condition: You must specify the identifier of the pipeline when</td>
</tr>
<tr>
<td></td>
<td>updating an existing pipeline with a new pipeline definition file.</td>
</tr>
<tr>
<td></td>
<td>Example: <code>--id df-00627471SOVYZEXAMPLE</code></td>
</tr>
</tbody>
</table>

## Common Options

For more information, see Common Options for AWS Data Pipeline Commands (p. 313).

**Output**

A response indicating that the new definition was successfully loaded, or, in the case where you are also using the `--create` command, an indication that the new pipeline was successfully activated.
Examples

The following examples show how to use `--put` to create a new pipeline (example one) and how to use
`--put` and `--id` to add a definition file to a pipeline (example two) or update a preexisting pipeline
definition file of a pipeline (example three).

```bash
datapipeline --create my-pipeline --put my-pipeline-definition.json
datapipeline --id df-00627471SOVYZEXAMPLE --put a-pipeline-definition.json
datapipeline --id df-00627471SOVYZEXAMPLE --put my-updated-pipeline-definition.json
```

Related Commands

- `--create` (p. 303)
- `--get, --g` (p. 305)

--rerun

Description

Reruns one or more specified objects from within a pipeline that is either currently running or has previously run. Resets the retry count of the object and then runs the object. It also tries to cancel the current attempt if an attempt is running.

Syntax

```bash
datapipeline --rerun object_id --id pipeline_id [Common Options]
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object_id</td>
<td>The identifier of the object.</td>
</tr>
<tr>
<td></td>
<td>Required: Yes</td>
</tr>
<tr>
<td></td>
<td>Example: o-06198791C436IEXAMPLE</td>
</tr>
<tr>
<td>--id pipeline_id</td>
<td>The identifier of the pipeline.</td>
</tr>
<tr>
<td></td>
<td>Required: Yes</td>
</tr>
<tr>
<td></td>
<td>Example: --id df-00627471SOVYZEXAMPLE</td>
</tr>
</tbody>
</table>
Common Options

For more information, see Common Options for AWS Data Pipeline Commands (p. 313).

Output

None. To see the status of the object set to rerun, use --list-runs.

Examples

Reruns the specified object in the indicated pipeline.

```
datapipeline --rerun o-06198791C436IEXAMPLE --id df-00627471SOVYZEXAMPLE
```

Related Commands

• --list-runs (p. 307)
• --list-pipelines (p. 307)

--validate

Description

Validates the pipeline definition for correct syntax. Also performs additional checks, such as a check for circular dependencies.

Syntax

```
datapipeline --validate pipeline_definition_file
```

Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pipeline_definition_file</td>
<td>The full path to the output file that receives the pipeline definition.</td>
</tr>
<tr>
<td></td>
<td>Default: standard output</td>
</tr>
<tr>
<td></td>
<td>Required: Yes</td>
</tr>
<tr>
<td></td>
<td>Example: my-pipeline.json</td>
</tr>
</tbody>
</table>
Common Options for AWS Data Pipeline Commands

Most of the AWS Data Pipeline commands support the following options.

--access-key aws_access_key
The access key ID associated with your AWS account.

If you specify --access-key, you must also specify --secret-key.

This option is required if you aren’t using a JSON credentials file (see --credentials).

Example: --access-key AKIAIOSFODNN7EXAMPLE

For CLI access, you need an access key ID and secret access key. Use IAM user access keys instead of AWS root account access keys. IAM lets you securely control access to AWS services and resources in your AWS account. For more information about creating access keys, see How Do I Get Security Credentials? in the AWS General Reference.

--credentials json_file
The location of the JSON file with your AWS credentials.

You don’t need to set this option if the JSON file is named credentials.json, and it exists in either your user home directory or the directory where the AWS Data Pipeline CLI is installed. The CLI automatically finds the JSON file if it exists in either location.

If you specify a credentials file (either using this option or by including credentials.json in one of its two supported locations), you don’t need to use the --access-key and --secret-key options.

For CLI access, you need an access key ID and secret access key. Use IAM user access keys instead of AWS root account access keys. IAM lets you securely control access to AWS services and resources in your AWS account. For more information about creating access keys, see How Do I Get Security Credentials? in the AWS General Reference.

--endpoint url
The URL of the AWS Data Pipeline endpoint that the CLI should use to contact the web service.

If you specify an endpoint both in a JSON file and with this command line option, the CLI ignores the endpoint set with this command line option.

--id pipeline_id
Use the specified pipeline identifier.

Example: --id df-00627471SOVYZEXAMPLE

--limit limit
The field limit for the pagination of objects.

--secret-key aws_secret_key
The secret access key associated with your AWS account.

If you specify --secret-key, you must also specify --access-key.

This option is required if you aren’t using a JSON credentials file (see --credentials).

Example: --secret-key wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

For CLI access, you need an access key ID and secret access key. Use IAM user access keys instead of AWS root account access keys. IAM lets you securely control access to AWS services and resources in your AWS account. For more information about creating access keys, see How Do I Get Security Credentials? in the AWS General Reference.
Creating a Pipeline Using the AWS Data Pipeline CLI

You can use the AWS Data Pipeline command line interface (CLI) to create and activate a pipeline. The example in this tutorial shows you how to copy data between two Amazon S3 buckets at a specific time interval.

Prerequisites

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

1. Install and configure the CLI. For more information, see Install the AWS Data Pipeline Command Line Interface (p. 295).
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, you must create these roles manually. For more information, see Setting Up IAM Roles (p. 4).

Tasks

Complete the following tasks.

1. Create a Pipeline Definition File (p. 314)
2. Activate the Pipeline (p. 315)

Create a Pipeline Definition File

First, define your activities and their data dependencies using a pipeline definition file. For the syntax and usage of pipeline definition files, see Pipeline Definition File Syntax (p. 53).

The following is the pipeline definition file for this example. For clarity, we've included only the required fields. We recommend that you use a text editor that can help you verify the syntax of JSON-formatted files and use the .json file extension.

```json
{
    "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"
  }
}]
}

Activate the Pipeline

You can create and activate your pipeline in a single step. 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.

To create your pipeline definition and activate your pipeline, use the following command.

```
datapipeline --create pipeline_name --put pipeline_file --activate --force
```
If your pipeline validates successfully, the command displays the following message:

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

Note the ID of your pipeline, because you'll use this value for most AWS Data Pipeline CLI commands. If the command fails, you'll see an error message. For information, see Troubleshooting (p. 153).

You can verify that your pipeline appears in the pipeline list using the following `--list-pipelines` (p. 307) command.

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

---

### Example Pipeline Definition Files

You can use the following example pipelines to quickly get started with AWS Data Pipeline.

**Example Pipelines**

- Copy Data from Amazon S3 to MySQL (p. 316)
- Extract Amazon S3 Data (CSV/TSV) to Amazon S3 using Hive (p. 318)
- Extract Amazon S3 Data (Custom Format) to Amazon S3 using Hive (p. 320)

For step-by-step instructions to create and use pipelines, read one or more of the detailed tutorials available in this guide. For more information, see Tutorials (p. 59).

**Copy Data from Amazon S3 to MySQL**

This example pipeline definition automatically creates an C2 instance that will copy the specified data from a CSV file in Amazon S3 into a MySQL database table. For simplicity, the structure of the example MySQL insert statement assumes that you have a CSV input file with two columns of data that you are writing into a MySQL database table that has two matching columns of the appropriate data type. If you have data of a different scope, modify the MySQL statement to include additional data columns or data types.

**Example Pipeline Definition**

```json
{
    "objects": [
        {
            "id": "Default",
            "logUri": "s3://testbucket/error_log",
            "schedule": {
                "ref": "MySchedule"
            }
        },
        {
            "id": "MySchedule",
            "type": "Schedule",
            "startDateTime": "2012-11-26T00:00:00",
```
This example has the following fields defined in the MySqlDataNode:

- **id**: User-defined identifier for the MySQL database, which is a label for your reference only.
- **type**: MySqlDataNode type that matches the kind of location for our data, which is an Amazon RDS instance using MySQL in this example.
table
Name of the database table that contains the data to copy. Replace table_name with the name of your database table.

username
User name of the database account that has sufficient permission to retrieve data from the database table. Replace user_name with the name of your user account.

"password"
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.

connectionString
JDBC connection string for CopyActivity to connect to the database.

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

### Extract Amazon S3 Data (CSV/TSV) to Amazon S3 using Hive

This example pipeline definition creates an Amazon EMR cluster to extract data from Apache web logs in Amazon S3 to a CSV file in Amazon S3 using Hive.

**Note**
You can accommodate tab-delimited (TSV) data files similarly to how this sample demonstrates using comma-delimited (CSV) files, if you change the reference to MyInputDataType and MyOutputDataType to be objects with a type "TSV" instead of "CSV".

#### Example Pipeline Definition

```
{

  "objects": [
    {
      "startDateTime": "2012-05-04T00:00:00",
      "id": "MyEmrResourcePeriod",
      "period": "1 day",
      "type": "Schedule",
      "endDateTime": "2012-05-05T00:00:00"
    },
    {
      "id": "MyHiveActivity",
      "maximumRetries": "5",
      "type": "HiveActivity",
      "schedule": {
        "ref": "MyEmrResourcePeriod"
      },
      "runsOn": {
        "ref": "MyEmrResource"
      },
      "input": {
        "ref": "MyInputData"
      },
      "output": {
        "ref": "MyOutputData"
      }
    }
  ]
}
```
"hiveScript": "INSERT OVERWRITE TABLE ${output1} select * from ${input1};"
Extract Amazon S3 Data (Custom Format) to Amazon S3 using Hive

This example pipeline definition creates an Amazon EMR cluster to extract data from Amazon S3 with Hive, using a custom file format specified by the columnSeparator and recordSeparator fields.

Example Pipeline Definition

```
{
  "objects": [
    {
      "startDateTime": "2012-05-04T00:00:00",
      "id": "MyEmrResourcePeriod",
      "period": "1 day",
      "type": "Schedule",
      "endDate": "2012-05-05T00:00:00"
    },
    {
      "id": "MyHiveActivity",
      "type": "HiveActivity",
      "schedule": {
        "ref": "MyEmrResourcePeriod"
      },
      "runsOn": {
        "ref": "MyEmrResource"
      },
      "input": {
        "ref": "MyInputData"
      },
      "output": {
        "ref": "MyOutputData"
      },
      "hiveScript": "INSERT OVERWRITE TABLE ${output1} select * from ${input1};"
    },
    {
      "id": "MyEmrResource",
      "type": "EmrCluster",
      "schedule": {
        "ref": "MyEmrResourcePeriod"
      },
      "masterInstanceType": "m1.small",
      "coreInstanceType": "m1.small",
      "enableDebugging": "true",
      "keyPair": "test-pair",
      "coreInstanceCount": "1",
      "actionOnTaskFailure": "continue",
      "maximumRetries": "1",
      "actionOnResourceFailure": "retryAll",
      "terminateAfter": "10 hour"
    }
  ]
}
```
API Version 2012-10-29

Web Service Limits

To ensure there is capacity for all users of the AWS Data Pipeline service, the web service imposes limits on the amount of resources you can allocate and the rate at which you can allocate them.

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.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Limit</th>
<th>Adjustable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of pipelines</td>
<td>100</td>
<td>Yes</td>
</tr>
<tr>
<td>Number of objects per pipeline</td>
<td>100</td>
<td>Yes</td>
</tr>
<tr>
<td>Number of active instances per object</td>
<td>5</td>
<td>Yes</td>
</tr>
<tr>
<td>Number of fields per object</td>
<td>50</td>
<td>No</td>
</tr>
<tr>
<td>Number of UTF8 bytes per field name or identifier</td>
<td>256</td>
<td>No</td>
</tr>
<tr>
<td>Number of UTF8 bytes per field</td>
<td>10,240</td>
<td>No</td>
</tr>
<tr>
<td>Number of UTF8 bytes per object</td>
<td>15,360 (including field names)</td>
<td>No</td>
</tr>
<tr>
<td>Rate of creation of a instance from an object</td>
<td>1 per 5 minutes</td>
<td>No</td>
</tr>
<tr>
<td>Retries of a pipeline activity</td>
<td>5 per task</td>
<td>No</td>
</tr>
<tr>
<td>Minimum delay between retry attempts</td>
<td>2 minutes</td>
<td>No</td>
</tr>
</tbody>
</table>
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 be 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.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Limit</th>
<th>Adjustable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum scheduling interval</td>
<td>15 minutes</td>
<td>No</td>
</tr>
<tr>
<td>Maximum number of roll-ups into a single object</td>
<td>32</td>
<td>No</td>
</tr>
<tr>
<td>Maximum number of EC2 instances per Ec2Resource object</td>
<td>1</td>
<td>No</td>
</tr>
</tbody>
</table>

### Web Service Call Limits

<table>
<thead>
<tr>
<th>API</th>
<th>Regular rate limit</th>
<th>Burst limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActivatePipeline</td>
<td>1 call per second</td>
<td>100 calls</td>
</tr>
<tr>
<td>CreatePipeline</td>
<td>1 call per second</td>
<td>100 calls</td>
</tr>
<tr>
<td>DeletePipeline</td>
<td>1 call per second</td>
<td>100 calls</td>
</tr>
<tr>
<td>DescribeObjects</td>
<td>2 calls per second</td>
<td>100 calls</td>
</tr>
<tr>
<td>DescribePipelines</td>
<td>1 call per second</td>
<td>100 calls</td>
</tr>
<tr>
<td>GetPipelineDefinition</td>
<td>1 call per second</td>
<td>100 calls</td>
</tr>
<tr>
<td>PollForTask</td>
<td>2 calls per second</td>
<td>100 calls</td>
</tr>
<tr>
<td>ListPipelines</td>
<td>1 call per second</td>
<td>100 calls</td>
</tr>
<tr>
<td>PutPipelineDefinition</td>
<td>1 call per second</td>
<td>100 calls</td>
</tr>
<tr>
<td>QueryObjects</td>
<td>2 calls per second</td>
<td>100 calls</td>
</tr>
<tr>
<td>ReportTaskProgress</td>
<td>10 calls per second</td>
<td>100 calls</td>
</tr>
</tbody>
</table>
## 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.

<table>
<thead>
<tr>
<th>API</th>
<th>Regular rate limit</th>
<th>Burst limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetTaskStatus</td>
<td>10 calls per second</td>
<td>100 calls</td>
</tr>
<tr>
<td>SetStatus</td>
<td>1 call per second</td>
<td>100 calls</td>
</tr>
<tr>
<td>ReportTaskRunnerHeartbeat</td>
<td>1 call per second</td>
<td>100 calls</td>
</tr>
<tr>
<td>ValidatePipelineDefinition</td>
<td>1 call per second</td>
<td>100 calls</td>
</tr>
</tbody>
</table>
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:

```json
{
  "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"
      },
      "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
  ]
}
```
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.
- **AWS Developer Tools** – Links to developer tools and resources that provide documentation, code samples, release notes, and other information to help you build innovative applications with AWS.
- **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:** November 25, 2014.

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Updated templates and console</td>
<td>Added new templates as reflected in the console. Updated the Getting Started chapter to use the <strong>Getting Started with ShellCommandActivity</strong> template. For more information, see Creating Pipelines Using Console Templates (p. 21).</td>
<td>25 November 2014</td>
</tr>
<tr>
<td>VPC support</td>
<td>Added support for launching resources into a virtual private cloud (VPC). For more information, see Launching Resources for Your Pipeline into a VPC (p. 46).</td>
<td>12 March 2014</td>
</tr>
<tr>
<td>Region support</td>
<td>Added support for multiple service regions. In addition to <code>us-east-1</code>, AWS Data Pipeline is supported in <code>eu-west-1</code>, <code>ap-northeast-1</code>, <code>ap-southeast-2</code>, and <code>us-west-2</code>.</td>
<td>20 February 2014</td>
</tr>
<tr>
<td>Redshift support</td>
<td>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. 131), RedshiftDataNode (p. 183), RedshiftDatabase (p. 277), and RedshiftCopyActivity (p. 227).</td>
<td>6 November 2013</td>
</tr>
<tr>
<td>PigActivity</td>
<td>Added PigActivity, which provides native support for Pig. For more information, see PigActivity (p. 218).</td>
<td>15 October 2013</td>
</tr>
<tr>
<td>New console template, activity, and data format</td>
<td>Added the new CrossRegion DynamoDB Copy console template, including the new HiveCopyActivity and DynamoDBExportDataFormat. For more information, see DynamoDB Cross Regional Table Copy (p. 23), HiveCopyActivity (p. 212), and DynamoDBExportDataFormat (p. 284).</td>
<td>21 August 2013</td>
</tr>
<tr>
<td>Cascading failures and reruns</td>
<td>Added information about AWS Data Pipeline cascading failure and rerun behavior. For more information, see Cascading Failures and Reruns (p. 51).</td>
<td>8 August 2013</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Release Date</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Troubleshooting video</td>
<td>Added the AWS Data Pipeline Basic Troubleshooting video. For more information, see Troubleshooting (p. 153).</td>
<td>17 July 2013</td>
</tr>
<tr>
<td>Editing active pipelines</td>
<td>Added more information about editing active pipelines and rerunning pipeline components. For more information, see Editing Your Pipelines (p. 40).</td>
<td>17 July 2013</td>
</tr>
<tr>
<td>Use resources in different regions</td>
<td>Added more information about using resources in different regions. For more information, see Using a Pipeline with Resources in Multiple Regions (p. 50).</td>
<td>17 June 2013</td>
</tr>
<tr>
<td>WAITING ON DEPENDENCIES status</td>
<td>CHECKING PRECONDITIONS status changed to WAITING_ON_DEPENDENCIES and added the @waitingOn runtime field for pipeline objects.</td>
<td>20 May 2013</td>
</tr>
<tr>
<td>DynamoDBDataFormat</td>
<td>Added DynamoDBDataFormat template.</td>
<td>23 April 2013</td>
</tr>
<tr>
<td>Process Web Logs video and Spot Instances support</td>
<td>Introduced the video &quot;Process Web Logs with AWS Data Pipeline, Amazon EMR, and Hive&quot;, and Amazon EC2 Spot Instances support.</td>
<td>21 February 2013</td>
</tr>
<tr>
<td>New Process Web Logs tutorial</td>
<td>Introduced a new tutorial (Process Access Logs Using Amazon EMR with Hive (p. 59)).</td>
<td>10 January 2013</td>
</tr>
<tr>
<td></td>
<td>The initial release of the AWS Data Pipeline Developer Guide.</td>
<td>20 December 2012</td>
</tr>
</tbody>
</table>