



Developer Guide

AWS Data Pipeline



API Version 2012-10-29

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Data Pipeline: Developer Guide

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

.....	ix
What is AWS Data Pipeline?	1
Migrating workloads from AWS Data Pipeline	2
Migrating workloads to AWS Glue	3
Migrating workloads to AWS Step Functions	3
Migrating workloads to Amazon MWAA	5
Mapping the concepts	6
Samples	7
Related services	8
Accessing AWS Data Pipeline	8
Pricing	9
Supported Instance Types for Pipeline Work Activities	9
Default Amazon EC2 Instances by AWS Region	10
Additional Supported Amazon EC2 Instances	11
Supported Amazon EC2 Instances for Amazon EMR Clusters	12
AWS Data Pipeline Concepts	14
Pipeline Definition	14
Pipeline Components, Instances, and Attempts	15
Task Runners	17
Data Nodes	18
Databases	19
Activities	19
Preconditions	20
System-Managed Preconditions	21
User-Managed Preconditions	21
Resources	21
Resource Limits	22
Supported Platforms	22
Amazon EC2 Spot Instances with Amazon EMR Clusters and AWS Data Pipeline	23
Actions	24
Proactively Monitoring Pipelines	24
Setting Up	25
Sign up for AWS	25
Sign up for an AWS account	25

Create a user with administrative access	26
Create IAM Roles for AWS Data Pipeline and Pipeline Resources	27
Allow IAM Principals (Users and Groups) to Perform Necessary Actions	27
Granting programmatic access	29
Getting Started with AWS Data Pipeline	31
Create the Pipeline	32
Monitor the Running Pipeline	33
View the Output	33
Delete the Pipeline	34
Working with pipelines	35
Creating a pipeline	35
Create a pipeline from Data Pipeline templates using the CLI	36
Viewing Your Pipelines	55
Interpreting Pipeline Status Codes	55
Interpreting Pipeline and Component Health State	57
Viewing Your Pipeline Definitions	58
Viewing Pipeline Instance Details	59
Viewing Pipeline Logs	60
Editing Your Pipeline	61
Limitations	62
Editing a Pipeline Using the AWS CLI	62
Cloning Your Pipeline	63
Tagging Your Pipeline	64
Deactivating Your Pipeline	65
Deactivate Your Pipeline Using the AWS CLI	65
Deleting Your Pipeline	66
Staging Data and Tables with Activities	66
Data Staging with ShellCommandActivity	68
Table Staging with Hive and Staging-Supported Data Nodes	69
Table Staging with Hive and Staging-Unsupported Data Nodes	70
Using Resources in Multiple Regions	72
Cascading failures and reruns	74
Activities	75
Data nodes and preconditions	75
Resources	75
Rerunning cascade-failed objects	75

Cascade-failure and backfills	75
Pipeline definition file syntax	76
File Structure	76
Pipeline Fields	77
User-defined fields	78
Working with the API	79
Install the AWS SDK	79
Making an HTTP Request to AWS Data Pipeline	80
Security	85
Data Protection	86
Identity and Access Management	87
IAM Policies for AWS Data Pipeline	88
Example Policies for AWS Data Pipeline	91
IAM Roles	95
Logging and Monitoring	99
AWS Data Pipeline Information in CloudTrail	99
Understanding AWS Data Pipeline Log File Entries	100
Incident Response	101
Compliance Validation	102
Resilience	102
Infrastructure Security	102
Configuration and Vulnerability Analysis in AWS Data Pipeline	102
Tutorials	103
Process Data Using Amazon EMR with Hadoop Streaming	103
Before You Begin	104
Using the CLI	104
Copy CSV Data from Amazon S3 to Amazon S3	108
Before You Begin	109
Using the CLI	110
Export MySQL Data to Amazon S3	117
Before You Begin	117
Using the CLI	119
Copy Data to Amazon Redshift	128
Before You Begin: Configure COPY Options	128
Before You Begin: Set up Pipeline, Security, and Cluster	129
Using the CLI	131

Pipeline Expressions and Functions	141
Simple Data Types	141
DateTime	141
Numeric	141
Object References	141
Period	142
String	142
Expressions	142
Referencing Fields and Objects	143
Nested Expressions	144
Lists	144
Node Expression	145
Expression Evaluation	146
Mathematical Functions	146
String Functions	147
Date and Time Functions	148
Special Characters	156
Pipeline Object Reference	158
Data Nodes	159
DynamoDBDataNode	160
MySQLDataNode	167
RedshiftDataNode	174
S3DataNode	182
SqlDataNode	189
Activities	196
CopyActivity	196
EmrActivity	204
HadoopActivity	214
HiveActivity	225
HiveCopyActivity	234
PigActivity	244
RedshiftCopyActivity	258
ShellCommandActivity	272
SqlActivity	281
Resources	289
Ec2Resource	290

EmrCluster	301
HttpProxy	332
Preconditions	335
DynamoDBDataExists	335
DynamoDBTableExists	339
Exists	343
S3KeyExists	347
S3PrefixNotEmpty	351
ShellCommandPrecondition	356
Databases	360
JdbcDatabase	361
RdsDatabase	363
RedshiftDatabase	365
Data Formats	367
CSV Data Format	368
Custom Data Format	369
DynamoDBFormat	371
DynamoDBExportFormat	374
RegEx Data Format	377
TSV Data Format	378
Actions	380
SnsAlarm	380
Terminate	382
Schedule	384
Examples	384
Syntax	389
Utilities	391
ShellScriptConfig	391
EmrConfiguration	393
Property	398
Working with Task Runner	401
Task Runner on AWS Data Pipeline-Managed Resources	401
Executing Work on Existing Resources Using Task Runner	403
Installing Task Runner	404
(Optional) Granting Task Runner Access to Amazon RDS	405
Starting Task Runner	407

Verifying Task Runner Logging	408
Task Runner Threads and Preconditions	408
Task Runner Configuration Options	408
Using Task Runner with a Proxy	411
Task Runner and Custom AMIs	411
Troubleshooting	413
Locating Errors in Pipelines	413
Identifying the Amazon EMR Cluster that Serves Your Pipeline	414
Interpreting Pipeline Status Details	414
Locating Error Logs	416
Pipeline Logs	416
Hadoop Job and Amazon EMR Step Logs	417
Resolving Common Problems	417
Pipeline Stuck in Pending Status	418
Pipeline Component Stuck in Waiting for Runner Status	418
Pipeline Component Stuck in WAITING_ON_DEPENDENCIES Status	419
Run Doesn't Start When Scheduled	420
Pipeline Components Run in Wrong Order	420
EMR Cluster Fails With Error: The security token included in the request is invalid	421
Insufficient Permissions to Access Resources	421
Status Code: 400 Error Code: PipelineNotFoundException	421
Creating a Pipeline Causes a Security Token Error	421
Cannot See Pipeline Details in the Console	421
Error in remote runner Status Code: 404, AWS Service: Amazon S3	422
Access Denied - Not Authorized to Perform Function datapipeline:	422
Older Amazon EMR AMIs May Create False Data for Large CSV Files	423
Increasing AWS Data Pipeline Limits	423
Limits	424
Account Limits	424
Web Service Call Limits	425
Scaling Considerations	427
AWS Data Pipeline Resources	428
Document History	430

AWS Data Pipeline is no longer available to new customers. Existing customers of AWS Data Pipeline can continue to use the service as normal. [Learn more](#)

What is AWS Data Pipeline?

Note

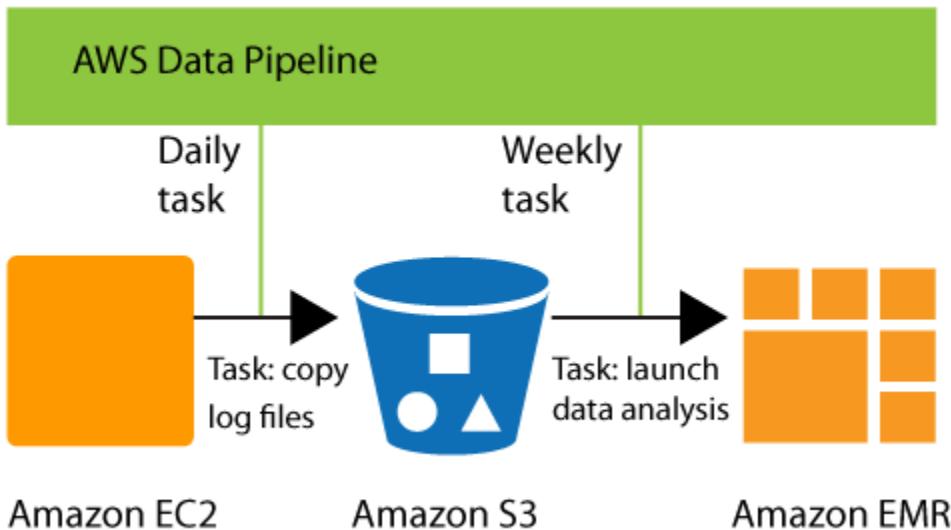
AWS Data Pipeline service is in maintenance mode and no new features or region expansions are planned. To learn more and to find out how to migrate your existing workloads, see [Migrating workloads from AWS Data Pipeline](#).

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

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

- A *pipeline definition* specifies the business logic of your data management. For more information, see [Pipeline definition file syntax](#).
- A *pipeline* schedules and runs tasks by creating Amazon EC2 instances to perform the defined work activities. You upload your pipeline definition to the pipeline, and then activate the pipeline. You can edit the pipeline definition for a running pipeline and activate the pipeline again for it to take effect. You can deactivate the pipeline, modify a data source, and then activate the pipeline again. When you are finished with your pipeline, you can delete it.
- *Task Runner* polls for tasks and then performs those tasks. For example, Task Runner could copy log files to Amazon S3 and launch Amazon EMR clusters. Task Runner is installed and runs automatically on resources created by your pipeline definitions. You can write a custom task runner application, or you can use the Task Runner application that is provided by AWS Data Pipeline. For more information, see [Task Runners](#).

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



Contents

- [Migrating workloads from AWS Data Pipeline](#)
- [Related services](#)
- [Accessing AWS Data Pipeline](#)
- [Pricing](#)
- [Supported Instance Types for Pipeline Work Activities](#)

Migrating workloads from AWS Data Pipeline

AWS launched the AWS Data Pipeline service in 2012. At that time, customers were looking for a service to help them reliably move data between different data sources using a variety of compute options. Now, there are other services that offer customers a better experience. For example, you can use AWS Glue to run and orchestrate Apache Spark applications, AWS Step Functions to help orchestrate AWS service components, or Amazon Managed Workflows for Apache Airflow (Amazon MWAA) to help manage workflow orchestration for Apache Airflow.

This topic explains how to migrate from AWS Data Pipeline to alternative options. The option you choose depends on your current workload on AWS Data Pipeline. You can migrate typical use cases of AWS Data Pipeline to either AWS Glue, AWS Step Functions, or Amazon MWAA.

Migrating workloads to AWS Glue

[AWS Glue](#) is a serverless data integration service that makes it easy for analytics users to discover, prepare, move, and integrate data from multiple sources. It includes tooling for authoring, running jobs, and orchestrating workflows. With AWS Glue, you can discover and connect to more than 70 diverse data sources and manage your data in a centralized data catalog. You can visually create, run, and monitor extract, transform, and load (ETL) pipelines to load data into your data lakes. Also, you can immediately search and query cataloged data using Amazon Athena, Amazon EMR, and Amazon Redshift Spectrum.

We recommend migrating your AWS Data Pipeline workload to AWS Glue when:

- You're looking for a serverless data integration service that supports various data sources, authoring interfaces including visual editors and notebooks, and advanced data management capabilities such as data quality and sensitive data detection.
- Your workload can be migrated to AWS Glue workflows, jobs (in Python or Apache Spark) and crawlers (for example, your existing pipeline is built on top of Apache Spark).
- You require a single platform that can handle all aspects of your data pipeline, including ingestion, processing, transfer, integrity testing, and quality checks.
- Your existing pipeline was created from a pre-defined template on the AWS Data Pipeline console, such as exporting a DynamoDB table to Amazon S3, and you are looking for the same purpose template.
- Your workload does not depend on a specific Hadoop ecosystem application like Apache Hive.
- Your workload does not require orchestrating on-premises servers.

AWS charges an hourly rate, billed by the second, for crawlers (discovering data) and ETL jobs (processing and loading data). AWS Glue Studio is a built-in orchestration engine for AWS Glue resources, and is offered at no additional cost. Learn more about pricing in [AWS Glue Pricing](#).

Migrating workloads to AWS Step Functions

[AWS Step Functions](#) is a serverless orchestration service that lets you build workflows for your business-critical applications. With Step Functions you use a visual editor to build workflows and integrate directly with over 11,000 actions for over 250 AWS services, such as AWS Lambda, Amazon EMR, DynamoDB and more. You can use Step Functions for orchestrating data processing pipelines, handling errors, and working with the throttling limits on the underlying AWS services.

You can create workflows that process and publish machine learning models, orchestrate micro-services, as well as control AWS services, such as AWS Glue, to create extract, transform, and load (ETL) workflows. You also can create long-running, automated workflows for applications that require human interaction.

Similarly to AWS Data Pipeline, AWS Step Functions is a fully managed service provided by AWS. You will not be required to manage infrastructure, patch workers, manage OS version updates or similar.

We recommend migrating your AWS Data Pipeline workload to AWS Step Functions when:

- You're looking for a serverless, highly available workflow orchestration service.
- You're looking for a cost-effective solution that charges at a granularity of a single task execution.
- Your workloads are orchestrating tasks for multiple other AWS services, such as Amazon EMR, Lambda, AWS Glue, or DynamoDB.
- You're looking for a low-code solution that comes with a drag-and-drop visual designer for workflow creation and does not require learning new programming concepts.
- You're looking for a service that provides integrations with over 250 other AWS services covering over 11,000 actions out-of-the-box, as well as allowing integrations with custom non-AWS services and activities.

Both AWS Data Pipeline and Step Functions use JSON format to define workflows. This allows to store your workflows in source control, manage versions, control access, and automate with CI/CD. Step Functions are using a syntax called Amazon State Language which is fully based on JSON, and allows a seamless transition between the textual and visual representations of the workflow.

With Step Functions, you can choose the same version of Amazon EMR that you're currently using in AWS Data Pipeline.

For migrating activities on AWS Data Pipeline managed resources, you can use [AWS SDK service integration](#) on Step Functions to automate resource provisioning and cleaning up.

For migrating activities on on-premises servers, user-managed EC2 instances, or a user-managed EMR cluster, you can install an [SSM agent](#) to the instance. You can initiate the command through the [AWS Systems Manager Run Command](#) from Step Functions. You can also initiate the state machine from the schedule defined in [Amazon EventBridge](#).

AWS Step Functions has two types of workflows: Standard Workflows and Express Workflows. For Standard Workflows, you're charged based on the number of state transitions required to run your application. For Express Workflows, you're charged based on the number of requests for your workflow and its duration. Learn more about pricing in [AWS Step Functions Pricing](#).

Migrating workloads to Amazon MWAA

[Amazon MWAA](#) (Managed Workflows for Apache Airflow) is a managed orchestration service for [Apache Airflow](#) that makes it easier to set up and operate end-to-end data pipelines in the cloud at scale. Apache Airflow is an open-source tool used to programmatically author, schedule, and monitor sequences of processes and tasks referred to as "workflows". With Amazon MWAA, you can use Airflow and Python programming language to create workflows without having to manage the underlying infrastructure for scalability, availability, and security. Amazon MWAA automatically scales its workflow execution capacity to meet your needs, and is integrated with AWS security services to help provide you with fast and secure access to your data.

Similarly to AWS Data Pipeline, Amazon MWAA is fully managed services provided by AWS. While you need to learn several new concepts specific to these services, you are not required to manage infrastructure, patch workers, manage OS version updates or similar.

We recommend migrating your AWS Data Pipeline workloads to Amazon MWAA when:

- You're looking for a managed, highly available service to orchestrate workflows written in Python.
- You want to transition to a fully-managed, widely-adopted open-source technology, Apache Airflow, for maximum portability.
- You require a single platform that can handle all aspects of your data pipeline, including ingestion, processing, transfer, integrity testing, and quality checks.
- You're looking for a service designed for data pipeline orchestration with features such as rich UI for observability, restarts for failed workflows, backfills, and retries for tasks.
- You're looking for a service that comes with more than 800 pre-built operators and sensors, covering AWS as well as non-AWS services.

Amazon MWAA workflows are defined as Directed Acyclic Graphs (DAGs) using Python, so you can also treat them as source code. Airflow's extensible Python framework enables you to build workflows connecting with virtually any technology. It comes with a rich user interface for viewing

and monitoring workflows and can be easily integrated with version control systems to automate the CI/CD process.

With Amazon MWAA, you can choose the same version of Amazon EMR that you're currently using in AWS Data Pipeline.

AWS charges for the time your Airflow environment runs plus any additional auto scaling to provide more worker or web server capacity. Learn more about pricing in [Amazon Managed Workflows for Apache Airflow Pricing](#).

Mapping the concepts

The following table contains mapping of major concepts used by the services. It will help people familiar with Data Pipeline to understand the Step Functions and MWAA terminology.

Data Pipeline	Glue	Step Functions	Amazon MWAA
Pipelines	Workflows	Workflows	Direct acyclic graphs
Pipeline definition JSON	Workflow definition or Python-based blueprints	Amazon State Language JSON	Python-based
Activities	Jobs	States and Tasks	Tasks (Operators and Sensors)
Instances	Job runs	Executions	DAG runs
Attempts	Retry attempts	Catchers and retriers	Retries
Pipeline schedule	Schedule triggers	EventBridge Scheduler tasks	Cron, timetables, data-aware
Pipeline expressions and functions	Blueprint library	Step Functions intrinsic functions and AWS Lambda	Extensible Python framework

Samples

The following sections lists public examples that you can refer to migrate from AWS Data Pipeline to individual services. You can refer them as examples, and build your own pipeline on the individual services by updating and testing it based on your use case.

AWS Glue samples

The following list contains sample implementations for the most common AWS Data Pipeline use-cases with AWS Glue.

- [Running Spark jobs](#)
- [Copying data from JDBC to Amazon S3](#) (including Amazon Redshift)
- [Copying data from Amazon S3 to JDBC](#) (including Amazon Redshift)
- [Copying data from Amazon S3 to DynamoDB](#)
- [Moving data to and from Amazon Redshift](#)
- [Cross-account cross-Region access to DynamoDB tables](#)

AWS Step Functions samples

The following list contains sample implementations for the most common AWS Data Pipeline use-cases with AWS Step Functions.

- [Managing an Amazon EMR job](#)
- [Running a data processing job on Amazon EMR Serverless](#)
- [Running Hive/Pig/Hadoop jobs](#)
- [Querying large datasets](#) (Amazon Athena, Amazon S3, AWS Glue)
- [Running ETL workflows using Amazon Redshift](#)
- [Orchestrating AWS Glue crawlers](#)

See additional [tutorials](#) and [samples projects](#) for using AWS Step Functions.

Amazon MWAA samples

The following list contains sample implementations for the most common AWS Data Pipeline use-cases with Amazon MWAA.

- [Running an Amazon EMR job](#)
- [Creating a custom plugin for Apache Hive and Hadoop](#)
- [Copying data from Amazon S3 to Redshift](#)
- [Executing a Shell script on a remote EC2 instance](#)
- [Orchestrating hybrid \(on-prem\) workflows](#)

See additional [tutorials](#) and [samples projects](#) for using Amazon MWAA.

Related services

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

- Amazon DynamoDB — Provides a fully managed NoSQL database with fast performance at a low cost. For more information, see [Amazon DynamoDB Developer Guide](#).
- Amazon RDS — Provides a fully managed relational database that scales to large datasets. For more information, see [Amazon Relational Database Service Developer Guide](#).
- Amazon Redshift — Provides a fast, fully managed, petabyte-scale data warehouse that makes it easy and cost-effective to analyze a vast amount of data. For more information, see [Amazon Redshift Database Developer Guide](#).
- Amazon S3 — Provides secure, durable, and highly scalable object storage. For more information, see [Amazon Simple Storage Service User Guide](#).

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

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

Accessing AWS Data Pipeline

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

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

Pricing

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

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

Supported Instance Types for Pipeline Work Activities

When AWS Data Pipeline runs a pipeline, it compiles the pipeline components to create a set of actionable Amazon EC2 instances. Each instance contains all the information for performing 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.

EC2 instances come in different configurations, which are known as *instance types*. Each instance type has a different CPU, input/output, and storage capacity. In addition to specifying the instance type for an activity, you can choose different purchasing options. Not all instance types are available in all AWS Regions. If an instance type is not available, your pipeline may fail to provision or may be stuck provisioning. For information about instance availability, see the [Amazon EC2](#)

[Pricing Page](#). Open the link for your instance purchasing option and filter by **Region** to see if an instance type is available in the Region. For more information about these instance types, families, and virtualization types, see [Amazon EC2 Instances](#) and [Amazon Linux AMI Instance Type Matrix](#).

The following tables describe the instance types that AWS Data Pipeline supports. You can use AWS Data Pipeline to launch Amazon EC2 instances in any Region, including Regions where AWS Data Pipeline is not supported. For information about Regions where AWS Data Pipeline is supported, see [AWS Regions and Endpoints](#).

Contents

- [Default Amazon EC2 Instances by AWS Region](#)
- [Additional Supported Amazon EC2 Instances](#)
- [Supported Amazon EC2 Instances for Amazon EMR Clusters](#)

Default Amazon EC2 Instances by AWS Region

If you do not specify an instance type in your pipeline definition, AWS Data Pipeline launches an instance by default.

The following table lists the Amazon EC2 instances that AWS Data Pipeline uses by default in those Regions where AWS Data Pipeline is supported.

Region Name	Region	Instance Type
US East (N. Virginia)	us-east-1	m1.small
US West (Oregon)	us-west-2	m1.small
Asia Pacific (Sydney)	ap-southeast-2	m1.small
Asia Pacific (Tokyo)	ap-northeast-1	m1.small
EU (Ireland)	eu-west-1	m1.small

The following table lists the Amazon EC2 instances that AWS Data Pipeline launches by default in those Regions where AWS Data Pipeline is not supported.

Region Name	Region	Instance Type
US East (Ohio)	us-east-2	t2.small
US West (N. California)	us-west-1	m1.small
Asia Pacific (Mumbai)	ap-south-1	t2.small
Asia Pacific (Singapore)	ap-southeast-1	m1.small
Asia Pacific (Seoul)	ap-northeast-2	t2.small
Canada (Central)	ca-central-1	t2.small
EU (Frankfurt)	eu-central-1	t2.small
EU (London)	eu-west-2	t2.small
EU (Paris)	eu-west-3	t2.small
South America (São Paulo)	sa-east-1	m1.small

Additional Supported Amazon EC2 Instances

In addition to the default instances that are created if you don't specify an instance type in your pipeline definition, the following instances are supported.

The following table lists the Amazon EC2 instances that AWS Data Pipeline supports and can create, if specified.

Instance Class	Instance Types
General purpose	t2.nano t2.micro t2.small t2.medium t2.large
Compute optimized	c3.large c3.xlarge c3.2xlarge c3.4xlarge c3.8xlarge c4.large c4.xlarge c4.2xlarge c4.4xlarge c4.8xlarge c5.xlarge c5.9xlarge c5.2xlarge c5.4xlarge c5.9xlarge c5.18xlarge c5d.xlarge c5d.2xlarge c5d.4xlarge c5d.9xlarge c5d.18xlarge

Instance Class	Instance Types
Memory optimized	m3.medium m3.large m3.xlarge m3.2xlarge m4.large m4.xlarge m4.2xlarge m4.4xlarge m4.10xlarge m4.16xlarge m5.xlarge m5.2xlarge m5.4xlarge m5.12xlarge m5.24xlarge m5d.xlarge m5d.2xlarge m5d.4xlarge m5d.12xlarge m5d.24xlarge r3.large r3.xlarge r3.2xlarge r3.4xlarge r3.8xlarge r4.large r4.xlarge r4.2xlarge r4.4xlarge r4.8xlarge r4.16xlarge
Storage optimized	i2.xlarge i2.2xlarge i2.4xlarge i2.8xlarge hs1.8xlarge g2.2xlarge g2.8xlarge d2.xlarge d2.2xlarge d2.4xlarge d2.8xlarge

Supported Amazon EC2 Instances for Amazon EMR Clusters

This table lists the Amazon EC2 instances that AWS Data Pipeline supports and can create for Amazon EMR clusters, if specified. For more information, see [Supported Instance Types](#) in the *Amazon EMR Management Guide*.

Instance Class	Instance Types
General purpose	m1.small m1.medium m1.large m1.xlarge m3.xlarge m3.2xlarge
Compute optimized	c1.medium c1.xlarge c3.xlarge c3.2xlarge c3.4xlarge c3.8xlarge cc1.4xlarge cc2.8xlarge c4.large c4.xlarge c4.2xlarge c4.4xlarge c4.8xlarge c5.xlarge c5.9xlarge c5.2xlarge c5.4xlarge c5.9xlarge c5.18xlarge c5d.xlarge c5d.2xlarge c5d.4xlarge c5d.9xlarge c5d.18xlarge
Memory optimized	m2.xlarge m2.2xlarge m2.4xlarge r3.xlarge r3.2xlarge r3.4xlarge r3.8xlarge cr1.8xlarge m4.large m4.xlarge m4.2xlarge m4.4xlarge m4.10xlarge m4.16xlarge m5.xlarge m5.2xlarge m5.4xlarge m5.12xlarge m5.24xlarge m5d.xlarge m5d.2xlarge m5d.4xlarge m5d.12xlarge

Instance Class	Instance Types
	m5d.24xlarge r4.large r4.xlarge r4.2xlarge r4.4xlarge r4.8xlarge r4.16xlarge
Storage optimized	h1.4xlarge hs1.2xlarge hs1.4xlarge hs1.8xlarge i2.xlarge i2.2xlarge i2.4large i2.8xlarge d2.xlarge d2.2xlarge d2.4xlarge d2.8xlarge
Accelerated computing	g2.2xlarge cg1.4xlarge

AWS Data Pipeline Concepts

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

Contents

- [Pipeline Definition](#)
- [Pipeline Components, Instances, and Attempts](#)
- [Task Runners](#)
- [Data Nodes](#)
- [Databases](#)
- [Activities](#)
- [Preconditions](#)
- [Resources](#)
- [Actions](#)

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

For example, in your pipeline definition, you might specify that log files generated by your application are archived each month in 2013 to an Amazon S3 bucket. AWS Data Pipeline would then create 12 tasks, each copying over a month's worth of data, regardless of whether the month contained 30, 31, 28, or 29 days.

You can create a pipeline definition in the following ways:

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

A pipeline definition can contain the following types of components.

Pipeline Components

Data Nodes

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

Activities

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

Preconditions

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

Resources

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

Actions

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](#).

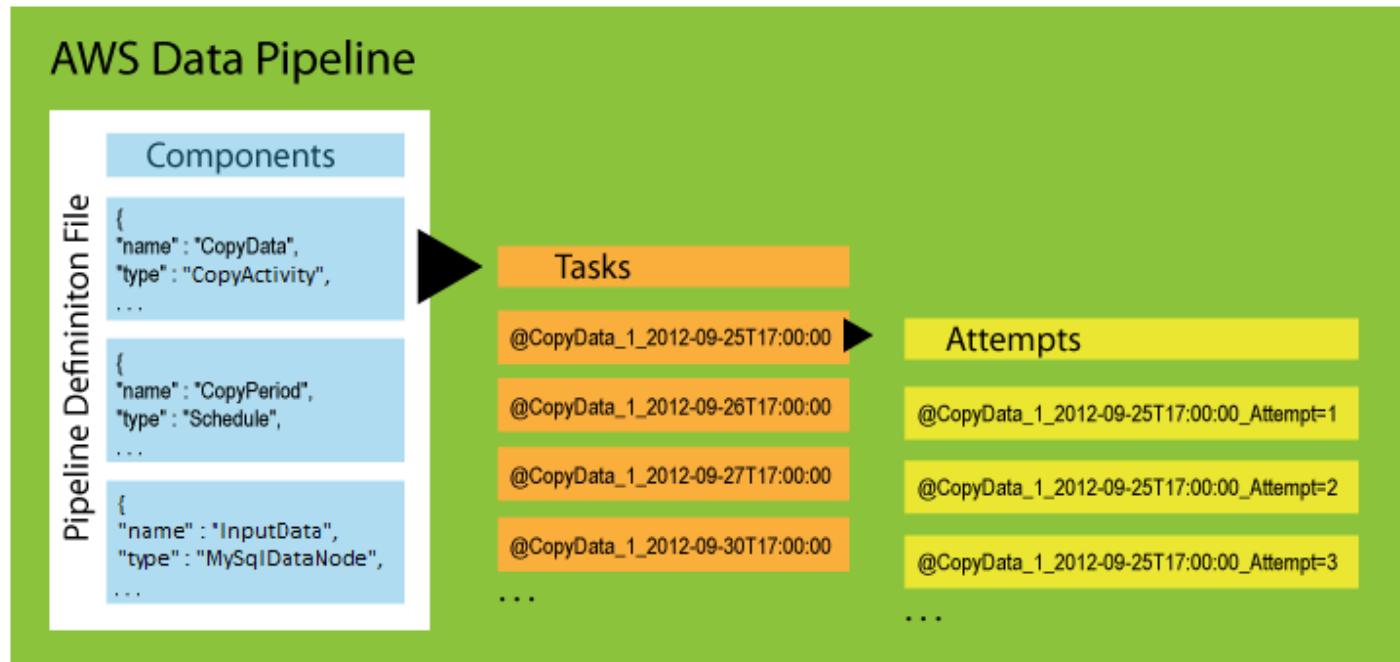
Pipeline Components, Instances, and Attempts

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

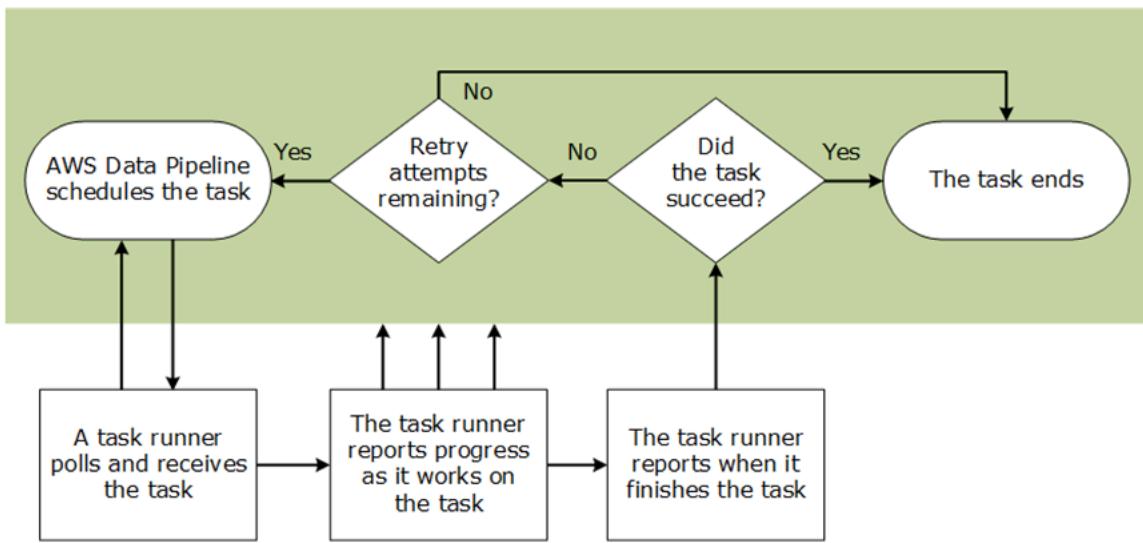


Task Runners

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

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

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



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

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

For more information about working with Task Runner, see [Working with Task Runner](#).

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](#)

A DynamoDB table that contains data for [HiveActivity](#) or [EmrActivity](#) to use.

[SqlDataNode](#)

An SQL table and database query that represent data for a pipeline activity to use.

Note

Previously, MySqlDataNode was used. Use SqlDataNode instead.

[RedshiftDataNode](#)

An Amazon Redshift table that contains data for [RedshiftCopyActivity](#) to use.

[S3DataNode](#)

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](#)

A JDBC database.

[RdsDatabase](#)

An Amazon RDS database.

[RedshiftDatabase](#)

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](#)

Copies data from one location to another.

[EmrActivity](#)

Runs an Amazon EMR cluster.

[HiveActivity](#)

Runs a Hive query on an Amazon EMR cluster.

[HiveCopyActivity](#)

Runs a Hive query on an Amazon EMR cluster with support for advanced data filtering and support for [S3DataNode](#) and [DynamoDBDataNode](#).

[PigActivity](#)

Runs a Pig script on an Amazon EMR cluster.

[RedshiftCopyActivity](#)

Copies data to and from Amazon Redshift tables.

[ShellCommandActivity](#)

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

[SqlActivity](#)

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](#).

Preconditions

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

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

System-Managed Preconditions

[DynamoDBDataExists](#)

Checks whether data exists in a specific DynamoDB table.

[DynamoDBTableExists](#)

Checks whether a DynamoDB table exists.

[S3KeyExists](#)

Checks whether an Amazon S3 key exists.

[S3PrefixNotEmpty](#)

Checks whether an Amazon S3 prefix is empty.

User-Managed Preconditions

[Exists](#)

Checks whether a data node exists.

[ShellCommandPrecondition](#)

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](#)

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

[EmrCluster](#)

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

Resources can run in the same region with their working dataset, even a region different than AWS Data Pipeline. For more information, see [Using a Pipeline with Resources in Multiple Regions](#).

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 one 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 can launch 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*.

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 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.

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 Instances. Spot Instances let you use spare EC2 instances and run them. 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 maximum price to Amazon EMR when your cluster is launched. It 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.

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:

- Your Spot Instances can terminate when the Spot Instance price goes above your maximum price for the instance, or due to Amazon EC2 capacity reasons. 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 termination.
- Spot Instances can take more time to start as they fulfill capacity asynchronously. Therefore, a Spot Instance 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 maximum price is too low.

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 `snsalarm` in the `onLateAction` field of `EmrActivity`.

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 SNS](#). In addition to SNS notifications, you can use the AWS Data Pipeline console and CLI to obtain pipeline status information.

AWS Data Pipeline supports the following actions:

SnsAlarm

An action that sends an SNS notification to a topic based on `onSuccess`, `OnFail`, and `onLateAction` events.

Terminate

An action that triggers the cancellation of a pending or unfinished activity, resource, or data node. You cannot terminate actions that include `onSuccess`, `OnFail`, or `onLateAction`.

Proactively Monitoring Pipelines

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

Setting up for AWS Data Pipeline

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

Tasks

- [Sign up for AWS](#)
- [Create IAM Roles for AWS Data Pipeline and Pipeline Resources](#)
- [Allow IAM Principals \(Users and Groups\) to Perform Necessary Actions](#)
- [Granting programmatic access](#)

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

Sign up for AWS

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

Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open <https://portal.aws.amazon.com/billing/signup>.
2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call or text message and entering a verification code on the phone keypad.

When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform [tasks that require root user access](#).

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to <https://aws.amazon.com/> and choosing **My Account**.

Create a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the [AWS Management Console](#) as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

For help signing in by using root user, see [Signing in as the root user](#) in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

For instructions, see [Enable a virtual MFA device for your AWS account root user \(console\)](#) in the *IAM User Guide*.

Create a user with administrative access

1. Enable IAM Identity Center.

For instructions, see [Enabling AWS IAM Identity Center](#) in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to a user.

For a tutorial about using the IAM Identity Center directory as your identity source, see [Configure user access with the default IAM Identity Center directory](#) in the *AWS IAM Identity Center User Guide*.

Sign in as the user with administrative access

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

For help signing in using an IAM Identity Center user, see [Signing in to the AWS access portal](#) in the *AWS Sign-In User Guide*.

Assign access to additional users

1. In IAM Identity Center, create a permission set that follows the best practice of applying least-privilege permissions.

For instructions, see [Create a permission set](#) in the *AWS IAM Identity Center User Guide*.

2. Assign users to a group, and then assign single sign-on access to the group.

For instructions, see [Add groups](#) in the *AWS IAM Identity Center User Guide*.

Create IAM Roles for AWS Data Pipeline and Pipeline Resources

AWS Data Pipeline requires IAM roles that determine the permissions to perform actions and access AWS resources. The *pipeline role* determines the permissions that AWS Data Pipeline has, and a *resource role* determines the permissions that applications running on pipeline resources, such as EC2 instances, have. You specify these roles when you create a pipeline. Even if you do not specify a custom role and use the default roles `DataPipelineDefaultRole` and `DataPipelineDefaultResourceRole`, you must first create the roles and attach permissions policies. For more information, see [IAM Roles for AWS Data Pipeline](#).

Allow IAM Principals (Users and Groups) to Perform Necessary Actions

To work with a pipeline, an IAM principal (a user or group) in your account must be allowed to perform required [AWS Data Pipeline actions](#) and actions for other services as defined by your pipeline.

To simplify permissions, the `AWSDataPipeline_FullAccess` managed policy is available for you to attach to IAM principals. This managed policy allows the principal to perform all actions that a user requires and the `iam:PassRole` action on the default roles used with AWS Data Pipeline when a custom role is not specified.

We highly recommend that you carefully evaluate this managed policy and restrict permissions only to those that your users require. If necessary, use this policy as a starting point, and then remove permissions to create a more restrictive inline permissions policy that you can attach to IAM principals. For more information and example permissions policies, see [Example Policies for AWS Data Pipeline](#)

A policy statement similar to the following example must be included in a policy attached to any IAM principal that uses the pipeline. This statement allows the IAM principal to perform the PassRole action on the roles that a pipeline uses. If you do not use default roles, replace *MyPipelineRole* and *MyResourceRole* with the custom roles that you create.

JSON

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

The following procedure demonstrates how to create an IAM group, attach the **AWSDataPipeline_FullAccess** managed policy to the group, and then add users to the group. You can use this procedure for any inline policy

To create a user group **DataPipelineDevelopers** and attach the **AWSDataPipeline_FullAccess** policy

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Groups**, **Create New Group**.
3. Enter a **Group Name**, for example, **DataPipelineDevelopers**, and then choose **Next Step**.
4. Enter **AWSDataPipeline_FullAccess** for **Filter** and then select it from the list.

5. Choose **Next Step** and then choose **Create Group**.
6. To add users to the group:
 - a. Select the group you created from the list of groups.
 - b. Choose **Group Actions, Add Users to Group**.
 - c. Select the users you want to add from the list and then choose **Add Users to Group**.

Granting programmatic access

Users need programmatic access if they want to interact with AWS outside of the AWS Management Console. The way to grant programmatic access depends on the type of user that's accessing AWS.

To grant users programmatic access, choose one of the following options.

Which user needs programmatic access?	To	By
IAM	(Recommended) Use console credentials as temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	<p>Following the instructions for the interface that you want to use.</p> <ul style="list-style-type: none">• For the AWS CLI, see Login for AWS local development in the <i>AWS Command Line Interface User Guide</i>.• For AWS SDKs, see Login for AWS local development in the <i>AWS SDKs and Tools Reference Guide</i>.
Workforce identity (Users managed in IAM Identity Center)	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	<p>Following the instructions for the interface that you want to use.</p> <ul style="list-style-type: none">• For the AWS CLI, see Configuring the AWS

Which user needs programmatic access?	To	By
		<p>CLI to use AWS IAM Identity Center in the <i>AWS Command Line Interface User Guide</i>.</p> <ul style="list-style-type: none"> For AWS SDKs, tools, and AWS APIs, see IAM Identity Center authentication in the <i>AWS SDKs and Tools Reference Guide</i>.
IAM	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions in Using temporary credentials with AWS resources in the <i>IAM User Guide</i> .
IAM	<p>(Not recommended)</p> <p>Use long-term credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.</p>	<p>Following the instructions for the interface that you want to use.</p> <ul style="list-style-type: none"> For the AWS CLI, see Authenticating using IAM user credentials in the <i>AWS Command Line Interface User Guide</i>. For AWS SDKs and tools, see Authenticate using long-term credentials in the <i>AWS SDKs and Tools Reference Guide</i>. For AWS APIs, see Managing access keys for IAM users in the <i>IAM User Guide</i>.

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](#) that define the work to perform, and [data nodes](#) that define the location and type of input and output data.

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

Prerequisites

Before you begin, complete the tasks in [Setting up for AWS Data Pipeline](#).

Pipeline Objects

The pipeline uses the following objects:

[ShellCommandActivity](#)

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

[S3DataNode](#) (input)

The S3 bucket that contains the input log file.

[S3DataNode](#) (output)

The S3 bucket for the output.

[Ec2Resource](#)

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](#)

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

Tasks

- [Create the Pipeline](#)
- [Monitor the Running Pipeline](#)
- [View the Output](#)
- [Delete the Pipeline](#)

Create the Pipeline

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

To create the pipeline

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

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

If you prefer, you can disable logging instead.

10. Under **Security/Access**, leave **IAM roles** set to **Default**.

11. Click **Activate**.

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

Monitor the Running Pipeline

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

To monitor the progress of your pipeline

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



Tip

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

2. When the status of every object in your pipeline is FINISHED, your pipeline has successfully completed the scheduled tasks.
3. If your pipeline doesn't complete successfully, check your pipeline settings for issues. For more information about troubleshooting failed or incomplete instance runs of your pipeline, see [Resolving Common Problems](#).

View the Output

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

Delete the Pipeline

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

To delete your pipeline

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

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

Working with pipelines

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

Important

Before you begin, see [Setting up for AWS Data Pipeline](#).

Contents

- [Creating a pipeline](#)
- [Viewing Your Pipelines](#)
- [Editing Your Pipeline](#)
- [Cloning Your Pipeline](#)
- [Tagging Your Pipeline](#)
- [Deactivating Your Pipeline](#)
- [Deleting Your Pipeline](#)
- [Staging Data and Tables with Pipeline Activities](#)
- [Using a Pipeline with Resources in Multiple Regions](#)
- [Cascading failures and reruns](#)
- [Pipeline definition file syntax](#)
- [Working with the API](#)

Creating a pipeline

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

- Use the AWS Command Line Interface (CLI) with a template provided for your convenience. For more information, see [Create a pipeline from Data Pipeline templates using the CLI](#).
- Use the AWS Command Line Interface (CLI) with a pipeline definition file in JSON format.
- Use an AWS SDK with a language-specific API. For more information, see [Working with the API](#).

Create a pipeline from Data Pipeline templates using the CLI

Data Pipeline provides several pre-configured pipeline definitions, known as templates. You can use templates to get started with AWS Data Pipeline quickly. These templates are available in a public bucket at the Amazon S3 location: `s3://datapipeline-us-east-1/templates/`. These predefined templates are created to achieve specific use cases and can be used to create pipelines. You can use `aws s3 ls --recursive "s3://datapipeline-us-east-1/templates/"` to list all the available templates.

Create a pipeline from a template using the CLI

Assume you want to create a pipeline that exports a DynamoDB table to Amazon S3. The template to be used in this case can be found at: `s3://datapipeline-us-east-1/templates/DynamoDB Templates/Export DynamoDB table to S3.json`.

To download the template JSON and create a pipeline using the CLI

1. Download the template using the `aws s3 cp` CLI or curl. For example:

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

2. Make changes to the downloaded template as needed. For example, to use latest EMR release version, change the `releaseLabel` field in `EmrClusterForBackup` object, change the master and core instance types, and change the default values of parameters in the template.
3. Create a pipeline using the `create-pipeline` CLI. For example:

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

4. Note the created pipeline ID.
5. Use `put-pipeline-definition` to upload the definition. Provide values of the parameters whose default values you want to override using the `--parameter-values` option.

For more information about templates, see [Choose a template](#).

Choose a template

The following templates are available for download from the Amazon S3 bucket: `s3://datipeline-us-east-1/templates/`.

Templates

- [Getting started using ShellCommandActivity](#)
- [Run AWS CLI command](#)
- [Export DynamoDB table to S3](#)
- [Import DynamoDB backup data from S3](#)
- [Run job on an Amazon EMR cluster](#)
- [Full copy of Amazon RDS MySQL Table to Amazon S3](#)
- [Incremental copy of Amazon RDS MySQL table to Amazon S3](#)
- [Load S3 data into Amazon RDS MySQL table](#)
- [Full copy of Amazon RDS MySQL table to Amazon Redshift](#)
- [Incremental copy of an Amazon RDS MySQL table to Amazon Redshift](#)
- [Load data from Amazon S3 into Amazon Redshift](#)

Getting started using ShellCommandActivity

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

The template uses the following pipeline objects:

- ShellCommandActivity
- S3InputNode
- S3OutputNode
- Ec2Resource

Run AWS CLI command

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

Export DynamoDB table to S3

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

The template uses the following pipeline objects:

- [EmrActivity](#)
- [EmrCluster](#)
- [DynamoDBDataNode](#)
- [S3DataNode](#)

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 are updated with those from the backup data and new items are added to the table. This template uses an Amazon EMR cluster, which is sized proportionally to the value of the throughput available to the DynamoDB table. Although you can increase IOPs on a table, this may incur additional costs while importing and exporting. Previously, import used a HiveActivity but now uses native MapReduce.

The template uses the following pipeline objects:

- [EmrActivity](#)
- [EmrCluster](#)
- [DynamoDBDataNode](#)
- [S3DataNode](#)
- [S3PrefixNotEmpty](#)

Run job on an Amazon EMR 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](#)
- [EmrCluster](#)

Full copy of Amazon RDS MySQL Table to Amazon 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](#)
- [Ec2Resource](#)
- [SqlDataNode](#)
- [S3DataNode](#)

Incremental copy of Amazon RDS MySQL table to Amazon 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 Amazon RDS MySQL table must have a Last Modified column.

This template copies changes that are made to the table between scheduled intervals starting from the scheduled start time. The schedule type is time series so if a copy was scheduled for a certain hour, AWS Data Pipeline copies the table rows that have a Last Modified time stamp that falls within the hour. Physical deletes to the table are not copied. The output is written in a timestamped subfolder under the Amazon S3 location on every scheduled run.

The template uses the following pipeline objects:

- [CopyActivity](#)
- [Ec2Resource](#)

- [SqlDataNode](#)
- [S3DataNode](#)

Load S3 data into Amazon 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 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 Amazon RDS MySQL table with those in the Amazon S3 data and adds new entries from the Amazon S3 data to the Amazon 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](#)
- [Ec2Resource](#)
- [SqlDataNode](#)
- [S3DataNode](#)

Amazon RDS to Amazon Redshift templates

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

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

If the Overwrite_Existing Amazon Redshift insert mode is being used:

- If a distribution key is not provided, a primary key on the Amazon RDS MySQL table is used.

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

For more information about Amazon Redshift, see the following topics:

- [Amazon Redshift cluster](#)
- Amazon Redshift [COPY](#)
- [Distribution styles](#) and DISTKEY [examples](#)
- [Sort Keys](#)

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

Data type translations between MySQL and Amazon Redshift

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

MySQL Data Type	Amazon Redshift Data Type	Notes
		number of digits may be specified in parentheses. Amazon Redshift: INT2. Signed two-byte integer
SMALLINT UNSIGNED, SMALLINT(size) UNSIGNED,	INTEGER	MySQL: 0 to 65535 UNSIGNED*. The maximum number of digits may be specified in parentheses Amazon Redshift: INT4. Signed four-byte integer
MEDIUMINT, MEDIUMINT(size)	INTEGER	MySQL: 388608 to 8388607. The maximum number of digits may be specified in parentheses Amazon Redshift: INT4. Signed four-byte integer
MEDIUMINT UNSIGNED, MEDIUMINT(size) UNSIGNED	INTEGER	MySQL: 0 to 16777215. The maximum number of digits may be specified in parentheses Amazon Redshift: INT4. Signed four-byte integer
INT, INT(size)	INTEGER	MySQL: 147483648 to 2147483647 Amazon Redshift: INT4. Signed four-byte integer

MySQL Data Type	Amazon Redshift Data Type	Notes
INT UNSIGNED, INT(size) UNSIGNED	BIGINT	MySQL: 0 to 4294967295 Amazon Redshift: INT8. Signed eight-byte integer
BIGINT BIGINT(size)	BIGINT	Amazon Redshift: INT8. Signed eight-byte integer
BIGINT UNSIGNED BIGINT(size) UNSIGNED	VARCHAR(20*4)	MySQL: 0 to 184467440 73709551615 Amazon Redshift: No native equivalent, so using char array.
FLOAT FLOAT(size,d) FLOAT(size,d) UNSIGNED	REAL	The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter . Amazon Redshift: FLOAT4
DOUBLE(size,d)	DOUBLE PRECISION	The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter . Amazon Redshift: FLOAT8

MySQL Data Type	Amazon Redshift Data Type	Notes
DECIMAL(size,d)	DECIMAL(size,d)	<p>A DOUBLE stored as a string, allowing for a fixed decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter.</p> <p>Amazon Redshift: No native equivalent.</p>
CHAR(size)	VARCHAR(size*4)	<p>Holds a fixed-length string, which can contain letters, numbers, and special characters. The fixed size is specified as the parameter in parentheses. Can store up to 255 characters.</p> <p>Right padded with spaces.</p> <p>Amazon Redshift: CHAR data type does not support multibyte character so VARCHAR is used.</p> <p>The maximum number of bytes per character is 4 according to RFC3629, which limits the character table to U+10FFFF.</p>

MySQL Data Type	Amazon Redshift Data Type	Notes
VARCHAR(size)	VARCHAR(size*4)	Can store up to 255 characters. VARCHAR does not support the following invalid UTF-8 code points: 0xD800- 0xDFFF, (Byte sequences: ED A0 80- ED BF BF), 0xFDD0- 0xFDEF, 0xFFFF, and 0xFFFF, (Byte sequences: EF B7 90- EF B7 AF, EF BF BE, and EF BF BF)
TINYTEXT	VARCHAR(255*4)	Holds a string with a maximum length of 255 characters
TEXT	VARCHAR(max)	Holds a string with a maximum length of 65,535 characters.
MEDIUMTEXT	VARCHAR(max)	0 to 16,777,215 Chars
LONGTEXT	VARCHAR(max)	0 to 4,294,967,295 Chars
BOOLEAN BOOL TINYINT(1)	BOOLEAN	MySQL: These types are synonyms for TINYINT(1) . A value of zero is considered false. Nonzero values are considered true.
BINARY[(M)]	varchar(255)	M is 0 to 255 bytes, FIXED
VARBINARY(M)	VARCHAR(max)	0 to 65,535 bytes
TINYBLOB	VARCHAR(255)	0 to 255 bytes
BLOB	VARCHAR(max)	0 to 65,535 bytes

MySQL Data Type	Amazon Redshift Data Type	Notes
MEDIUMBLOB	VARCHAR(max)	0 to 16,777,215 bytes
LONGBLOB	VARCHAR(max)	0 to 4,294,967,295 bytes
ENUM	VARCHAR(255*2)	The limit is not on the length of the literal enum string, but rather on the table definition for number of enum values.
SET	VARCHAR(255*2)	Like enum.
DATE	DATE	(YYYY-MM-DD) "1000-01-01" to "9999-12-31"
TIME	VARCHAR(10*4)	(hh:mm:ss) "-838:59:59" to "838:59:59"
DATETIME	TIMESTAMP	(YYYY-MM-DD hh:mm:ss) "1000-01-01 00:00:00" to "9999-12-31 23:59:59"
TIMESTAMP	TIMESTAMP	(YYYYMMDDhhmmss) 19700101000000 to 2037+
YEAR	VARCHAR(4*4)	(YYYY) 1900 to 2155

MySQL Data Type	Amazon Redshift Data Type	Notes
column SERIAL	<p>ID generation / This attribute is not needed for an OLAP data warehouse since this column is copied.</p> <p>SERIAL keyword is not added while translating.</p>	<p>SERIAL is in fact an entity named SEQUENCE. It exists independently on the rest of your table.</p> <p>column GENERATED BY DEFAULT</p> <p>equivalent to:</p> <pre>CREATE SEQUENCE name; CREATE TABLE table (column INTEGER NOT NULL DEFAULT nextval(name));</pre>
column BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE	<p>ID generation / This attribute is not needed for OLAP data warehouse since this column is copied.</p> <p>So SERIAL keyword is not added while translating.</p>	<p>SERIAL is in fact an entity named SEQUENCE. It exists independently on the rest of your table.</p> <p>column GENERATED BY DEFAULT</p> <p>equivalent to:</p> <pre>CREATE SEQUENCE name; CREATE TABLE table (column INTEGER NOT NULL DEFAULT nextval(name));</pre>

MySQL Data Type	Amazon Redshift Data Type	Notes
ZEROFILL	ZEROFILL keyword is not added while translating.	<p>INT UNSIGNED ZEROFILL NOT NULL</p> <p>ZEROFILL pads the displayed value of the field with zeros up to the display width specified in the column definition. Values longer than the display width are not truncated. Note that usage of ZEROFILL also implies UNSIGNED.</p>

Full copy of Amazon RDS MySQL table to Amazon Redshift

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

The template uses the following pipeline objects:

- [CopyActivity](#)
- [RedshiftCopyActivity](#)
- [S3DataNode](#)
- [SqlDataNode](#)
- [RedshiftDataNode](#)
- [RedshiftDatabase](#)

Incremental copy of an Amazon RDS MySQL table to Amazon Redshift

The **Incremental copy of Amazon RDS MySQL table to Amazon Redshift** template copies data from an Amazon RDS MySQL table to an Amazon Redshift table by staging data in an Amazon S3 folder.

The Amazon S3 staging folder must be in the same region as the Amazon Redshift cluster.

AWS Data Pipeline uses a translation script to create an Amazon Redshift table with the same schema as the source Amazon RDS MySQL table if it does not already exist. You must provide any Amazon RDS MySQL to Amazon Redshift column data type overrides you would like to apply during Amazon Redshift table creation.

This template copies changes that are made to the Amazon RDS MySQL table between scheduled intervals, starting from the scheduled start time. Physical deletes to the Amazon RDS MySQL table are not copied. You must provide the column name that stores the last modified time value.

When you use the default template to create pipelines for incremental Amazon RDS copy, an activity with the default name RDSToS3CopyActivity is created. You can rename it.

The template uses the following pipeline objects:

- [CopyActivity](#)
- [RedshiftCopyActivity](#)
- [S3DataNode](#)
- [SqlDataNode](#)
- [RedshiftDataNode](#)
- [RedshiftDatabase](#)

Load data from Amazon S3 into Amazon Redshift

The **Load data from S3 into Redshift** template copies data from an Amazon S3 folder into an Amazon 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 Amazon Redshift COPY options. The Amazon Redshift table must have the same schema as the data in Amazon S3. For COPY options, see [COPY](#) in the Amazon Redshift *Database Developer Guide*.

The template uses the following pipeline objects:

- [CopyActivity](#)
- [RedshiftCopyActivity](#)
- [S3DataNode](#)
- [RedshiftDataNode](#)
- [RedshiftDatabase](#)
- [Ec2Resource](#)

Creating a pipeline Using parametrized templates

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

Contents

- [Add myVariables to the pipeline definition](#)
- [Define parameter objects](#)
- [Define Parameter Values](#)
- [Submitting the pipeline definition](#)

Add myVariables to the pipeline definition

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

 **Note**

A pipeline definition has an upper limit of 50 parameters.

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

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

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

Define parameter objects

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

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

Note

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

The following table describes the attributes for parameter objects.

Parameter attributes

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

Attribute	Type	Description
		value for this parameter using parameter values, it overrides the default value.
isArray	Boolean	Indicates whether the parameter is an array.

Define Parameter Values

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

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

Submitting the pipeline definition

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

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

Note

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

Viewing Your Pipelines

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

To view your pipelines using the AWS CLI

- Use the following [list-pipelines](#) command to list your pipelines:

```
aws datapipeline list-pipelines
```

Interpreting Pipeline Status Codes

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

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

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

Status Codes

ACTIVATING

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

CANCELED

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

CASCADE_FAILED

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

DEACTIVATING

The pipeline is being deactivated.

FAILED

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

FINISHED

The component completed its assigned work.

INACTIVE

The pipeline was deactivated.

PAUSED

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

PENDING

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

RUNNING

The resource is running and ready to receive work.

SCHEDULED

The resource is scheduled to run.

SHUTTING_DOWN

The resource is shutting down after successfully completing its work.

SKIPPED

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

TIMEDOUT

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

VALIDATING

The pipeline definition is being validated by AWS Data Pipeline.

WAITING_FOR_RUNNER

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

WAITING_ON_DEPENDENCIES

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

Interpreting Pipeline and Component Health State

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

Pipeline Health States

HEALTHY

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

ERROR

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

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

No health status was reported for this pipeline.

Note

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

Component Health States

HEALTHY

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

ERROR

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

No Completed Executions or No Health Information Available

No health status was reported for this component.

Viewing Your Pipeline Definitions

Use the command line interface (CLI) to view your pipeline definition. 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](#).

When 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.

When 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-00627471S0VYZEXAMPLE
```

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-00627471S0VYZEXAMPLE
```

Viewing Pipeline Instance Details

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

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 large.

The following example retrieves information for all runs.

```
aws datapipeline list-runs --pipeline-id df-00627471S0VYZEXAMPLE
```

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

```
aws datapipeline list-runs --pipeline-id df-00627471S0VYZEXAMPLE --status finished
```

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

```
aws datapipeline list-runs --pipeline-id df-00627471S0VYZEXAMPLE --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) follow the above pipeline logging structure.

To view logs for a given pipeline run:

1. Retrieve the `ObjectId` by calling `query-objects` to get the exact object ID. For example:

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

`query-objects` is a paginated CLI and may return a pagination token if there are more executions for the given `pipeline-id`. You can use the token to go through all the attempts until you find the expected object. For example, a returned `ObjectId` would look like: `@TableBackupActivity_2023-05-020T18:05:18_Attempt=1`.

2. Using the `ObjectId`, retrieve the log location using:

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

Error message of a failed activity

To get the error message, first get the ObjectId using query-objects.

After retrieving the failed ObjectId, use the describe-objects CLI to get the actual error message.

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

Cancel or rerun or mark as finished an object

Use the set-status CLI to cancel a running object, or re-run a failed object or mark a running object as Finished.

First, get the object ID using the query-objects CLI. For example:

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

Use the set-status CLI to change the status of the desired object. For example:

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

Editing Your Pipeline

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

Contents

- [Limitations](#)

- [Editing a Pipeline Using the AWS CLI](#)

Limitations

While the pipeline is in the PENDING state and is not activated, you can't make any changes to it. After you activate a pipeline, you can edit the pipeline with the following restrictions. The changes you make apply to new runs of the pipeline objects after you save them and then activate the pipeline again.

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

Editing a Pipeline Using the AWS CLI

You can edit a pipeline using the command line tools.

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

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

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

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

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

```
aws datapipeline activate-pipeline --pipeline-id df-00627471S0VYZEXAMPLE
```

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

```
aws datapipeline activate-pipeline --pipeline-id df-00627471S0VYZEXAMPLE --start-timestamp YYYY-MM-DDTHH:MM:SSZ
```

To re-run one or more pipeline components, use the [set-status](#) command.

Cloning Your Pipeline

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

To clone a pipeline using the AWS CLI:

1. Create a new pipeline with a new name and unique ID. Note the returned pipeline ID.
2. Use the `get-pipeline-definition` CLI to get the pipeline definition of the existing pipeline to be cloned and write it to a temporary file. Note the absolute path of the file.
3. Use the `put-pipeline-definition` CLI to copy the pipeline definition from the existing pipeline to the new pipeline.
4. Use the `get-pipeline-definition` CLI to get the definition of the new pipeline to verify the pipeline definition.

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

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

# Put pipeline definition to new pipeline
```

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

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

Tagging Your Pipeline

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

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

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

To tag your pipeline using the AWS CLI

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

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

To add tags to an existing pipeline, use the [add-tags](#) command as follows:

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

To remove tags from an existing pipeline, use the [remove-tags](#) command as follows:

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

Deactivating Your Pipeline

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

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

When you activate a deactivated pipeline, you can specify when it resumes. Using the AWS CLI or the API, the pipeline resumes from the last completed execution by default, or you can specify the date and time to resume the pipeline.

Deactivate Your Pipeline Using the AWS CLI

Use the following [deactivate-pipeline](#) command to deactivate a pipeline:

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

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

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

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

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

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

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

Deleting Your Pipeline

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

Important

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

To delete a pipeline using the AWS CLI

To delete a pipeline, use the [delete-pipeline](#) command. The following command deletes the specified pipeline.

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

Staging Data and Tables with Pipeline Activities

AWS Data Pipeline can stage input and output data in your pipelines to make it easier to use certain activities, such as `ShellCommandActivity` and `HiveActivity`.

Data staging enables you to copy 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](#).

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.

Note

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

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

```
"schedule": {  
    "ref": "MySchedule"  
},  
"directoryPath": "s3://my_bucket/destination/#{format(@scheduledStartTime, 'YYYY-MM-  
dd_HHmmss')}"  
}  
,  
...  
}
```

Table Staging with Hive and Staging-Supported Data Nodes

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

 **Note**

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

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

```
"id": "MyInputData",
"type": "S3DataNode",
"schedule": {
    "ref": "MySchedule"
},
"directoryPath": "s3://test-hive/input"
},
{
"id": "MyOutputData",
"type": "S3DataNode",
"schedule": {
    "ref": "MySchedule"
},
"directoryPath": "s3://test-hive/output"
}
},
...
}
```

Table Staging with Hive and Staging-Unsupported Data Nodes

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

 **Note**

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

```
{
"id": "MyHiveActivity",
"type": "HiveActivity",
```

```
"schedule": {
    "ref": "MySchedule"
},
"runsOn": {
    "ref": "MyEmrResource"
},
"input": {
    "ref": "MyDynamoData"
},
"output": {
    "ref": "MyS3Data"
},
"hiveScript": "-- Map DynamoDB Table
SET dynamodb.endpoint=dynamodb.us-east-1.amazonaws.com;
SET dynamodb.throughput.read.percent = 0.5;
CREATE EXTERNAL TABLE dynamodb_table (item map<string,string>)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES ("dynamodb.table.name" = "#{input.tableName}");
INSERT OVERWRITE TABLE ${output1} SELECT * FROM dynamodb_table;"
},
{
    "id": "MyDynamoData",
    "type": "DynamoDBDataNode",
    "schedule": {
        "ref": "MySchedule"
    },
    "tableName": "MyDDBTable"
},
{
    "id": "MyS3Data",
    "type": "S3DataNode",
    "schedule": {
        "ref": "MySchedule"
    },
    "directoryPath": "s3://test-hive/output"
}
},
...
}
```

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 in a specified region, you also have the flexibility to co-locate your resources with their dependent datasets 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 Europe (Ireland) region, assuming that a large amount of data for the cluster to work on exists in the same region. In this example, the only difference from a typical pipeline is that the `EmrCluster` has a `region` field value set to `eu-west-1`.

```
{
  "objects": [
    {
      "id": "Hourly",
      "type": "Schedule",
      "startDateTime": "2014-11-19T07:48:00",
      "endDateTime": "2014-11-21T07:48:00",
      "period": "1 hours"
    },
    {
      "id": "MyCluster",
      "type": "EmrCluster",
      "masterInstanceType": "m3.medium",
      "region": "eu-west-1",
      "schedule": {
        "ref": "Hourly"
      }
    },
    {
      "id": "MyEmrActivity",
      "type": "EmrActivity",
      "schedule": {
        "ref": "Hourly"
      },
      "runsOn": {
        "ref": "MyCluster"
      }
    }
  ]
}
```

```
    },
    "step": "/home/hadoop/contrib/streaming/hadoop-streaming.jar,-input,s3n://
elasticmapreduce/samples/wordcount/input,-output,s3://eu-west-1-bucket/wordcount/
output/#{@scheduledStartTime},-mapper,s3n://elasticmapreduce/samples/wordcount/
wordSplitter.py,-reducer,aggregate"
}
]
}
```

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

 **Note**

The following list includes regions in which AWS Data Pipeline can orchestrate workflows and launch Amazon EMR or Amazon EC2 resources. AWS Data Pipeline may not be supported in these regions. For information about regions in which AWS Data Pipeline is supported, see [AWS Regions and Endpoints](#).

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

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

Cascading failures and reruns

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

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

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

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

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

Activities

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

Data nodes and preconditions

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

Resources

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

Rerunning cascade-failed objects

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

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

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

Cascade-failure and backfills

If you enable cascade failure and have a pipeline that creates many backfills, pipeline runtime errors can cause resources to be created and deleted in rapid succession without performing useful work. AWS Data Pipeline attempts to alert you about this situation with the following warning

message when you save a pipeline: *Pipeline_object_name* has 'failureAndRerunMode' field set to 'cascade' and you are about to create a backfill with scheduleStartTime *start_time*. This can result in rapid creation of pipeline objects in case of failures. This happens because cascade failure can quickly set downstream activities as CASCADE_FAILED and shut down 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](#).

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 need, add them to the pipeline definition file, and then add the appropriate fields. For more information about pipeline objects, see [Pipeline Object Reference](#).

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](#). For more information about functions that you can use to evaluate field values, see [Expression Evaluation](#).

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](#).

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

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

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

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

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

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

User-defined fields

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

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

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

 **Note**

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

Working with the API

 **Note**

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

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 simplifies calling the web service APIs from your preferred programming environment. For more information, see [Install the AWS SDK](#).

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: <>service>>_<>API version>>. <>operationName>>

For example, `DataPipeline_20121129.ActivatePipeline`

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

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

```
POST / HTTP/1.1
host: https://datapipeline.us-east-1.amazonaws.com
x-amz-date: Mon, 12 Nov 2012 17:49:52 GMT
x-amz-target: DataPipeline_20121129.ActivatePipeline
Authorization: AuthParams
Content-Type: application/x-amz-json-1.1
Content-Length: 39
Connection: Keep-Alive
```

HTTP Body Content

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

Format the Body of an HTTP Request

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

```
{
  "pipelineId": "df-00627471SOVYZEXAMPLE",
  "pipelineObjects": [
    {
      "id": "Default",
      "name": "Default",
      "slots": [
        {
          "key": "workerGroup",
```

```
        "stringValue": "MyWorkerGroup"
    ],
},
{"id": "Schedule",
"name": "Schedule",
"slots":
[
    {"key": "startDateTime",
     "stringValue": "2012-09-25T17:00:00"},
    {"key": "type",
     "stringValue": "Schedule"},
    {"key": "period",
     "stringValue": "1 hour"},
    {"key": "endDateTime",
     "stringValue": "2012-09-25T18:00:00"}
],
},
{"id": "SayHello",
"name": "SayHello",
"slots":
[
    {"key": "type",
     "stringValue": "ShellCommandActivity"},
    {"key": "command",
     "stringValue": "echo hello"},
    {"key": "parent",
     "refValue": "Default"},
    {"key": "schedule",
     "refValue": "Schedule"}
],
}
]
```

Handle the HTTP Response

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

- **HTTP/1.1**—This header is followed by a status code. A code value of 200 indicates a successful operation. Any other value indicates an error.

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

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

Sample AWS Data Pipeline JSON Request and Response

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

HTTP POST Request

```
POST / HTTP/1.1
host: https://datipeline.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"}
```

Security in AWS Data Pipeline

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The [shared responsibility model](#) describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the [AWS Compliance Programs](#). To learn about the compliance programs that apply to AWS Data Pipeline, see [AWS Services in Scope by Compliance Program](#).
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using AWS Data Pipeline. The following topics show you how to configure AWS Data Pipeline to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your AWS Data Pipeline resources.

Topics

- [Data Protection in AWS Data Pipeline](#)
- [Identity and Access Management for AWS Data Pipeline](#)
- [Logging and Monitoring in AWS Data Pipeline](#)
- [Incident Response in AWS Data Pipeline](#)
- [Compliance Validation for AWS Data Pipeline](#)
- [Resilience in AWS Data Pipeline](#)
- [Infrastructure Security in AWS Data Pipeline](#)
- [Configuration and Vulnerability Analysis in AWS Data Pipeline](#)

Data Protection in AWS Data Pipeline

The AWS [shared responsibility model](#) applies to data protection in AWS Data Pipeline. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the [Data Privacy FAQ](#). For information about data protection in Europe, see the [AWS Shared Responsibility Model and GDPR](#) blog post on the [AWS Security Blog](#).

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We recommend TLS 1.2 or later.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see [Federal Information Processing Standard \(FIPS\) 140-2](#).
- AWS Data Pipeline supports IMDSv2 for Amazon EMR and Amazon EC2 resources. To use IMDSv2 with Amazon EMR, use versions 5.23.1, 5.27.1, or 5.32 or later or version 6.2 or later. For more information, see [Configure metadata service requests to Amazon EC2 instances](#) and [Use IMDSv2](#).

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with AWS Data Pipeline or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Identity and Access Management for AWS Data Pipeline

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

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

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

AWS Data Pipeline is integrated with AWS Identity and Access Management (IAM), which offers a wide range of features:

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

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

Contents

- [IAM Policies for AWS Data Pipeline](#)
- [Example Policies for AWS Data Pipeline](#)
- [IAM Roles for AWS Data Pipeline](#)

IAM Policies for AWS Data Pipeline

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

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

Contents

- [Policy Syntax](#)
- [Controlling Access to Pipelines Using Tags](#)
- [Controlling Access to Pipelines Using Worker Groups](#)

Policy Syntax

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

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

The following elements make up a policy statement:

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

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

- `datapipeline:PipelineCreator` — To grant access to the user that created the pipeline. For an example, see [Grant the pipeline owner full access](#).
- `datapipeline:Tag` — To grant access based on pipeline tagging. For more information, see [Controlling Access to Pipelines Using Tags](#).
- `datapipeline:workerGroup` — To grant access based on the name of the worker group. For more information, see [Controlling Access to Pipelines Using Worker Groups](#).

Controlling Access to Pipelines Using Tags

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

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

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

To achieve this scenario, the manager tags the production pipelines with the "environment=production" tag and attaches the following policy to the developer IAM group. The

first statement grants read-only access to all pipelines. The second statement grants read/write access to pipelines that do not have an "environment=production" tag.

JSON

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

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

JSON

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

```
        "Resource": "*"
    }
]
```

For more examples, see [Grant users read-only access based on a tag](#) and [Grant users full access based on a tag](#).

Controlling Access to Pipelines Using Worker Groups

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

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

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

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

Example Policies for AWS Data Pipeline

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

Contents

- [Example 1: Grant users read-only access based on a tag](#)
- [Example 2: Grant users full access based on a tag](#)

- [Example 3: Grant the pipeline owner full access](#)
- [Example 4: Grant users access to the AWS Data Pipeline console](#)

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

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

The ListPipelines API action does not support tag-based authorization.

JSON

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

Example 2: Grant users full access based on a tag

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

JSON

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

Example 3: Grant the pipeline owner full access

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

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

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

This policy includes the action for PassRole permissions for specific resources tied to the roleARN that AWS Data Pipeline needs. For more information about the identity-based (IAM) PassRole permission, see the blog post [Granting Permission to Launch EC2 Instances with IAM Roles \(PassRole Permission\)](#).

JSON

```
{
```

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

IAM Roles for AWS Data Pipeline

AWS Data Pipeline uses AWS Identity and Access Management roles. The permissions policies attached to IAM roles determine what actions AWS Data Pipeline and your applications can perform, and what AWS resources they can access. For more information, see [IAM roles in the IAM User Guide](#).

AWS Data Pipeline requires two IAM roles:

- **The pipeline role** controls AWS Data Pipeline access to your AWS resources. In pipeline object definitions, the `role` field specifies this role.
- **The EC2 instance role** controls the access that applications running on EC2 instances, including the EC2 instances in Amazon EMR clusters, have to AWS resources. In pipeline object definitions, the `resourceRole` field specifies this role.

Important

If you created a pipeline before October 3, 2022 using the AWS Data Pipeline console with default roles, AWS Data Pipeline created the `DataPipelineDefaultRole` for you and attached the `AWSDataPipelineRole` managed policy to the role. As of October 3, 2022, the `AWSDataPipelineRole` managed policy is deprecated and the pipeline role must be specified for a pipeline when using the console.

We recommend that you review existing pipelines and determine if the `DataPipelineDefaultRole` is associated with the pipeline and whether the `AWSDataPipelineRole` is attached to that role. If so, review the access that this policy allows to ensure it is appropriate for your security requirements. Add, update, or replace the policies and policy statements attached to this role as necessary. Alternatively, you can update a pipeline to use a role that you create with different permissions policies.

Example Permissions Policies for AWS Data Pipeline Roles

Each role has one or more permissions policies attached to it that determine the AWS resources that the role can access and the actions that the role can perform. This topic provides an example permissions policy for the pipeline role. It also provides the contents of the `AmazonEC2RoleforDataPipelineRole`, which is the managed policy for the default EC2 instance role, `DataPipelineDefaultResourceRole`.

Example Pipeline Role Permissions Policy

The example policy that follows is scoped to allow essential functions that AWS Data Pipeline requires to run a pipeline with Amazon EC2 and Amazon EMR resources. It also provides permissions to access other AWS resources, such as Amazon Simple Storage Service and Amazon Simple Notification Service, that many pipelines require. If the objects defined in a pipeline do not require the resources of an AWS service, we strongly recommend that you remove permissions to access that service. For example, if your pipeline does not define a [DynamoDBDataNode](#) or use the [SnsAlarm](#) action, we recommend that you remove the allow statements for those actions.

- Replace **11112222333** with your AWS account ID.
 - Replace **NameOfDataPipelineRole** with the name of pipeline role (the role to which this policy is attached).
 - Replace **NameOfDataPipelineResourceRole** with the name of EC2 instance role.
 - Replace **us-west-1** with the appropriate Region for your application.

Default Managed Policy for the EC2 Instance Role

The contents of the `AmazonEC2RoleforDataPipelineRole` is shown below. This is the managed policy attached to the default resource role for AWS Data Pipeline, `DataPipelineDefaultResourceRole`. When you define a resource role for your pipeline, we recommend that you begin with this permissions policy and then remove permissions for AWS service actions that are not required.

Version 3 of the policy is shown, which is the most recent version at the time of this writing. View the most recent version of the policy using the IAM console.

JSON

```
"elasticmapreduce:AddJobFlowSteps",
"elasticmapreduce:Describe*",
"elasticmapreduce>ListInstance*",
"elasticmapreduce:ModifyInstanceGroups",
"rds:Describe*",
"redshift:DescribeClusters",
"redshift:DescribeClusterSecurityGroups",
"s3:*",
"sdb:*",
"sns:*",
"sqs:"

],
"Resource": ["*"]
}
}
```

Creating IAM Roles for AWS Data Pipeline and Editing Role Permissions

Use the following procedures to create roles for AWS Data Pipeline using the IAM console. The process consists of two steps. First, you create a permissions policy to attach to the role. Next, you create the role and attach the policy. After you create a role, you can change the role's permissions by attaching and detaching permissions policies.

 **Note**

When you create roles for AWS Data Pipeline using the console as described below, IAM creates and attaches the appropriate trust policies that the role requires.

To create a permissions policy to use with a role for AWS Data Pipeline

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Policies**, and then choose **Create policy**.
3. Choose the **JSON** tab.
4. If you are creating a pipeline role, copy and paste the contents of the policy example in [Example Pipeline Role Permissions Policy](#), editing it as appropriate for your security requirements. Alternatively, if you are creating a custom EC2 instance role, do the same for the example in [Default Managed Policy for the EC2 Instance Role](#).

5. Choose **Review policy**.
6. Enter a name for the policy—for example, **MyDataPipelineRolePolicy**—and an optional **Description**, and then choose **Create policy**.
7. Note the name of the policy. You need it when you create your role.

To create an IAM role for AWS Data Pipeline

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Roles**, and then choose **Create Role**.
3. Under **Choose a use case**, choose **Data Pipeline**.
4. Under **Select your use case**, do one of the following:
 - Choose **Data Pipeline** to create a pipeline role.
 - Choose **EC2 Role for Data Pipeline** to create a resource role.
5. Choose **Next: Permissions**.
6. If the default policy for AWS Data Pipeline is listed, proceed with the following steps to create the role and then edit it according to the instructions in the next procedure. Otherwise, enter the name of the policy that you created in the procedure above, and then select it from the list.
7. Choose **Next: Tags**, enter any tags to add to the role, and then choose **Next: Review**.
8. Enter a name for the role—for example, **MyDataPipelineRole**—and an optional **Description**, and then choose **Create role**.

To attach or detach a permissions policy for an IAM role for AWS Data Pipeline

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Roles**.
3. In the search box, begin typing the name of the role you want to edit—for example, **DataPipelineDefaultRole** or **MyDataPipelineRole**—and then choose the **Role name** from the list.
4. On the **Permissions** tab, do the following:
 - To detach a permissions policy, under **Permissions policies**, choose the remove button on the far right of the policy entry. Choose **Detach** when prompted to confirm.

- To attach a policy that you created earlier, choose **Attach policies**. In the search box, begin typing the name of the policy you want to edit, select it from the list, and then choose **Attach policy**.

Changing Roles for an Existing Pipelines

If you want to assign a different pipeline role or resource role to a pipeline, you can use the architect editor in the AWS Data Pipeline console.

To edit the roles assigned to a pipeline using the console

1. Open the AWS Data Pipeline console at <https://console.aws.amazon.com/datapipeline/>.
2. Select the pipeline from the list, and then choose **Actions**, **Edit**.
3. In the right pane of the architect editor, choose **Others**.
4. From the **Resource Role** and **Role** lists, choose the roles for AWS Data Pipeline that you want to assign, and then choose **Save**.

Logging and Monitoring in AWS Data Pipeline

AWS Data Pipeline is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS Data Pipeline. CloudTrail captures all API calls for AWS Data Pipeline as events. The calls captured include calls from the AWS Data Pipeline console and code calls to the AWS Data Pipeline API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS Data Pipeline. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to AWS Data Pipeline, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the [AWS CloudTrail User Guide](#).

AWS Data Pipeline Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in AWS Data Pipeline, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see [Viewing Events with CloudTrail Event History](#).

For an ongoing record of events in your AWS account, including events for AWS Data Pipeline, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- [Overview for Creating a Trail](#)
- [CloudTrail Supported Services and Integrations](#)
- [Configuring Amazon SNS Notifications for CloudTrail](#)
- [Receiving CloudTrail Log Files from Multiple Regions](#) and [Receiving CloudTrail Log Files from Multiple Accounts](#)

All of the AWS Data Pipeline actions are logged by CloudTrail 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 event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or IAM role credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the [CloudTrail userIdentity Element](#).

Understanding AWS Data Pipeline Log File Entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the **CreatePipeline** operation:

```
{  
  "Records": [  
    {  
      "eventVersion": "1.02",  
      "userIdentity": {  
        "type": "Root",  
        "principalId": "123456789012",  
        "arn": "arn:aws:iam::aws-account-id:role/role-name",  
        "accountId": "role-account-id",  
        "accessKeyId": "role-access-key"  
      },  
      "eventTime": "2014-11-13T19:15:15Z",  
      "eventSource": "datapipeline.amazonaws.com",  
      "eventName": "CreatePipeline",  
      "awsRegion": "us-east-1",  
      "sourceIPAddress": "72.21.196.64",  
      "userAgent": "aws-cli/1.5.2 Python/2.7.5 Darwin/13.4.0",  
      "requestParameters": {  
        "name": "testpipeline",  
        "uniqueId": "sounique"  
      },  
      "responseElements": {  
        "pipelineId": "df-06372391ZG65EXAMPLE"  
      },  
      "requestID": "65cbf1e8-6b69-11e4-8816-cfcbad04c45",  
      "eventID": "9f99dce0-0864-49a0-bffa-f72287197758",  
      "eventType": "AwsApiCall",  
      "recipientAccountId": "role-account-id"  
    },  
    ...additional entries  
  ]  
}
```

Incident Response in AWS Data Pipeline

Incident response for AWS Data Pipeline is an AWS responsibility. AWS has a formal, documented policy and program that governs incident response.

AWS operational issues with broad impact are posted on the AWS Service Health Dashboard. Operational issues are also posted to individual accounts via the Personal Health Dashboard.

Compliance Validation for AWS Data Pipeline

AWS Data Pipeline is not in scope of any AWS compliance programs. For a list of AWS services in scope of specific compliance programs, see [AWS Services in Scope by Compliance Program](#). For general information, see [AWS Compliance Programs](#).

Resilience in AWS Data Pipeline

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see [AWS Global Infrastructure](#).

Infrastructure Security in AWS Data Pipeline

As a managed service, AWS Data Pipeline is protected by the AWS global network security procedures that are described in the [Amazon Web Services: Overview of Security Processes](#) whitepaper.

You use AWS published API calls to access AWS Data Pipeline through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the [AWS Security Token Service](#) (AWS STS) to generate temporary security credentials to sign requests.

Configuration and Vulnerability Analysis in AWS Data Pipeline

Configuration and IT controls are a shared responsibility between AWS and you, our customer. For more information, see the AWS [shared responsibility model](#).

Tutorials

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

Tutorials

- [Process Data Using Amazon EMR with Hadoop Streaming](#)
- [Copy CSV Data Between Amazon S3 Buckets Using AWS Data Pipeline](#)
- [Export MySQL Data to Amazon S3 Using AWS Data Pipeline](#)
- [Copy Data to Amazon Redshift Using AWS Data Pipeline](#)

Process Data Using Amazon EMR with Hadoop Streaming

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

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

Pipeline Objects

The pipeline uses the following objects:

[EmrActivity](#)

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

[EmrCluster](#)

Resource AWS Data Pipeline uses to perform this activity.

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

Schedule

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

SnsAlarm

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

Contents

- [Before You Begin](#)
- [Launch a Cluster Using the Command Line](#)

Before You Begin

Be sure you've completed the following steps.

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

Launch a Cluster Using the Command Line

If you regularly run an Amazon EMR cluster to analyze web logs or perform analysis of scientific data, you can use AWS Data Pipeline to manage your Amazon EMR clusters. With AWS Data Pipeline, you can specify preconditions that must be met before the cluster is launched (for example, ensuring that today's data been uploaded to Amazon S3.) This tutorial walks you through launching a cluster that can be a model for a simple Amazon EMR-based pipeline, or as part of a more involved pipeline.

Prerequisites

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

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

Tasks

- [Creating the Pipeline Definition File](#)
- [Uploading and Activating the Pipeline Definition](#)
- [Monitor the Pipeline Runs](#)

Creating the Pipeline Definition File

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

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

```
{  
  "objects": [  
    {  
      "id": "Hourly",  
      "type": "Schedule",  
      "startDateTime": "2012-11-19T07:48:00",  
      "endDateTime": "2012-11-21T07:48:00",  
      "period": "1 hours"  
    },  
    {  
      "id": "MyCluster",  
      "type": "Cluster",  
      "name": "MyCluster",  
      "actions": [  
        {  
          "id": "RunStreamingJob",  
          "actionType": "RunHadoopStreamingJob",  
          "jobParameters": {  
            "scriptBootstrap": "s3://elasticmapreduce/libs/hadoop-mapreduce-examples/hadoop-mapreduce-examples.jar",  
            "script": "WordCount",  
            "args": ["hdfs://localhost:9000/user/hadoop/input", "hdfs://localhost:9000/user/hadoop/output"]  
          }  
        }  
      ]  
    }  
  ]  
}
```

```
"type": "EmrCluster",
"masterInstanceType": "m1.small",
"schedule": {
    "ref": "Hourly"
},
],
{
    "id": "MyEmrActivity",
    "type": "EmrActivity",
    "schedule": {
        "ref": "Hourly"
    },
    "runsOn": {
        "ref": "MyCluster"
    },
    "step": "/home/hadoop/contrib/streaming/hadoop-streaming.jar,-input,s3n://elasticmapreduce/samples/wordcount/input,-output,s3://myawsbucket/wordcount/output/#{@scheduledStartTime},-mapper,s3n://elasticmapreduce/samples/wordcount/wordSplitter.py,-reducer,aggregate"
}
]
```

This pipeline has three objects:

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

Uploading and Activating the Pipeline Definition

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

AWS CLI

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

```
aws datapipeline create-pipeline --name pipeline_name --unique-id token
{
    "pipelineId": "df-00627471S0VYZEXAMPLE"
}
```

To upload your pipeline definition, use the following [put-pipeline-definition](#) command.

```
aws datapipeline put-pipeline-definition --pipeline-id df-00627471S0VYZEXAMPLE --
pipeline-definition file://MyEmrPipelineDefinition.json
```

If your 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-00627471S0VYZEXAMPLE
```

You can verify that your pipeline appears in the pipeline list using the following [list-pipelines](#) command.

```
aws datapipeline list-pipelines
```

Monitor the Pipeline Runs

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

To check the progress of clusters launched by AWS Data Pipeline

1. Open the Amazon EMR console.

2. The clusters that were spawned by AWS Data Pipeline have a name formatted as follows:
`<pipeline-identifier>_@<emr-cluster-name>_<launch-time>`.

Elastic MapReduce Cluster List				
Create cluster		View details	Clone	Terminate
Filter: All clusters		Filter clusters ...	3 clusters (all loaded)	
Name	ID	Status		
<input type="checkbox"/> df-00592868ZT33HX1F5I0_@MyCluster_2014-06-29T02:00:00	j-20XJRAX6Z5HC4	Running		
<input type="checkbox"/> df-00592868ZT33HX1F5I0_@MyCluster_2014-06-29T01:00:00	j-32CYSLG57E6YT	Running		

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.

Upload	Create Folder	Actions
All Buckets / js-s3-bucket / wordcount		
Name		
<input type="checkbox"/>	2014-06-29T00:00:00	
<input type="checkbox"/>	2014-06-29T01:00:00	
<input type="checkbox"/>	2014-06-29T02:00:00	

Copy CSV Data Between Amazon S3 Buckets Using AWS Data Pipeline

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

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

Pipeline Objects

The pipeline uses the following objects:

[CopyActivity](#)

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

⚠ Important

There are limitations when using the CSV file format with CopyActivity and S3DataNode. For more information, see [CopyActivity](#).

[Schedule](#)

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

[Ec2Resource](#)

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

[S3DataNode](#)

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

[SnsAlarm](#)

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

Contents

- [Before You Begin](#)
- [Copy CSV Data Using the Command Line](#)

Before You Begin

Be sure you've completed the following steps.

- Complete the tasks in [Setting up for AWS Data Pipeline](#).
- (Optional) Set up a VPC for the instance and a security group for the VPC.
- Create an Amazon S3 bucket as a data source.

For more information, see [Create a Bucket](#) in the *Amazon Simple Storage Service User 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 User Guide*.

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

Copy CSV Data Using the Command Line

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

Prerequisites

Before you begin, you must complete the following steps:

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

Tasks

- [Define a Pipeline in JSON Format](#)
- [Upload and Activate the Pipeline Definition](#)

Define a Pipeline in JSON Format

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

Note

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

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

```
{  
  "objects": [  
    {  
      "id": "MySchedule",  
      "type": "Schedule",  
      "startDateTime": "2013-08-18T00:00:00",  
      "endDateTime": "2013-08-19T00:00:00",  
      "period": "1 day"  
    },  
    {  
      "id": "S3Input",  
      "type": "S3DataNode",  
      "schedule": {  
        "ref": "MySchedule"  
      },  
      "filePath": "s3://amzn-s3-demo-bucket/source/inputfile.csv"  
    },  
    {  
      "id": "S3Output",  
      "type": "S3DataNode",  
      "schedule": {  
        "ref": "MySchedule"  
      },  
      "filePath": "s3://amzn-s3-demo-bucket/destination/outputfile.csv"  
    },  
    {  
      "id": "MyEC2Resource",  
    }  
  ]  
}
```

```
"type": "Ec2Resource",
"schedule": {
    "ref": "MySchedule"
},
"instanceType": "m1.medium",
"role": "DataPipelineDefaultRole",
"resourceRole": "DataPipelineDefaultResourceRole"
},
{
    "id": "MyCopyActivity",
    "type": "CopyActivity",
    "runsOn": {
        "ref": "MyEC2Resource"
    },
    "input": {
        "ref": "S3Input"
    },
    "output": {
        "ref": "S3Output"
    },
    "schedule": {
        "ref": "MySchedule"
    }
}
]
```

Schedule

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

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

Amazon S3 Data Nodes

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

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

Id

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

Type

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

Schedule

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

Path

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

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

```
{  
  "id": "S3Output",
```

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

Resource

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

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

Id

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

Type

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

Schedule

The schedule on which to create this computational resource.

instanceType

The size of the EC2 instance to create. Ensure that you set the appropriate size of EC2 instance that best matches the load of the work that you want to perform with AWS Data Pipeline.

In this case, we set an m1.medium EC2 instance. For more information about the different instance types and when to use each one, see [Amazon EC2 Instance Types](#) topic at <http://aws.amazon.com/ec2/instance-types/>.

Role

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

resourceRole

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

Activity

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

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

Id

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

Type

The type of activity to perform, such as MyCopyActivity.

runsOn

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

Input

The location of the data to copy.

Output

The target location data.

Schedule

The schedule on which to run this activity.

Upload and Activate the Pipeline Definition

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

AWS CLI

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

```
aws datapipeline create-pipeline --name pipeline_name --unique-id token
{
    "pipelineId": "df-00627471S0VYZEXAMPLE"
}
```

To upload your pipeline definition, use the following [put-pipeline-definition](#) command.

```
aws datapipeline put-pipeline-definition --pipeline-id df-00627471S0VYZEXAMPLE --
pipeline-definition file://MyEmrPipelineDefinition.json
```

If your 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-00627471S0VYZEXAMPLE
```

You can verify that your pipeline appears in the pipeline list using the following [list-pipelines](#) command.

```
aws datapipeline list-pipelines
```

Export MySQL Data to Amazon S3 Using AWS Data Pipeline

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

Pipeline Objects

The pipeline uses the following objects:

- [CopyActivity](#)
- [Ec2Resource](#)
- [MySqlDataNode](#)
- [S3DataNode](#)
- [SnsAlarm](#)

Contents

- [Before You Begin](#)
- [Copy MySQL Data Using the Command Line](#)

Before You Begin

Be sure you've completed the following steps.

- Complete the tasks in [Setting up for AWS Data Pipeline](#).
- (Optional) Set up a VPC for the instance and a security group for the VPC.
- Create an Amazon S3 bucket as a data output.

For more information, see [Create a Bucket](#) in *Amazon Simple Storage Service User Guide*.

- Create and launch a MySQL database instance as your data source.

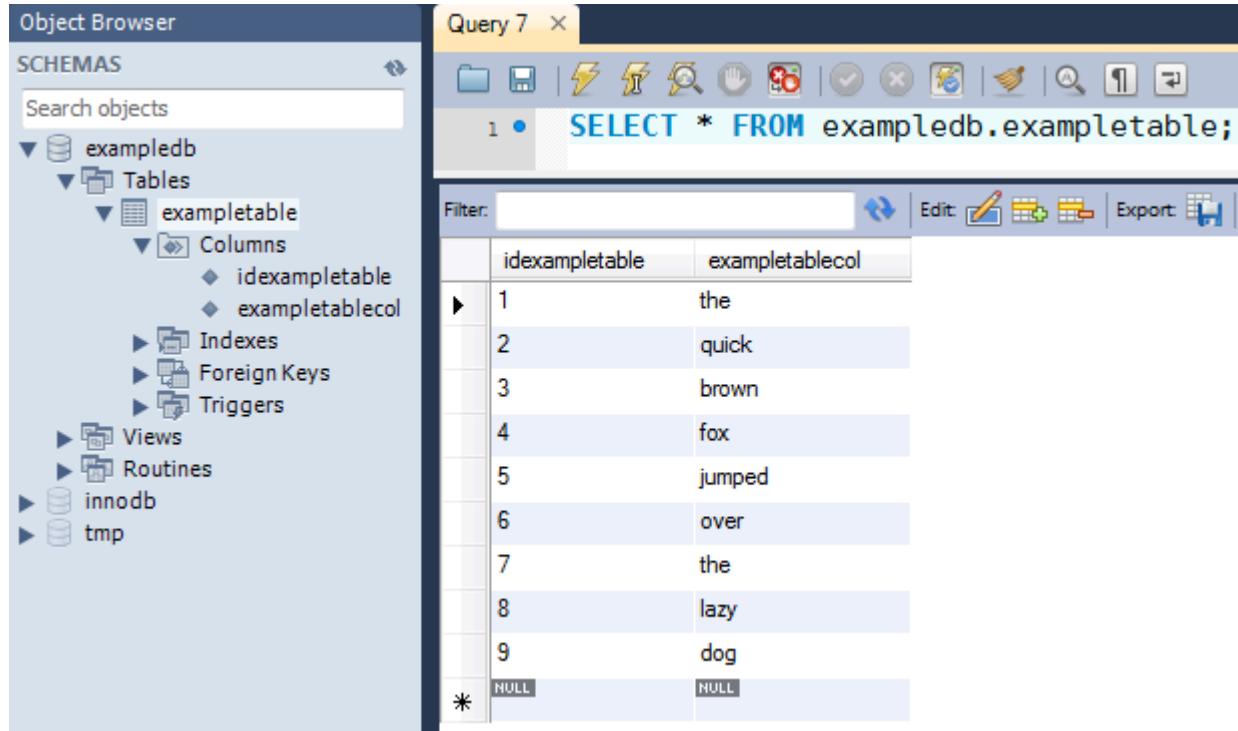
For more information, see [Launch a DB Instance](#) in the *Amazon RDS Getting Started Guide*. After you have an Amazon RDS instance, see [Create a Table](#) in the MySQL documentation.

Note

Make a note of the user name and the password you used for creating the MySQL instance. After you've launched your MySQL database instance, make a note of the instance's endpoint. You'll need this information later.

- Connect to your MySQL database instance, create a table, and then add test data values to the newly created table.

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



The screenshot shows the MySQL Workbench interface. On the left is the Object Browser, which displays the schema structure of the 'exampledb' database. It shows a 'Tables' node with a single child table named 'examptable'. This table has two columns: 'idexamptable' and 'examptablecol'. The table contains 9 rows of data. On the right is the Query Editor window titled 'Query 7'. It contains the SQL query: 'SELECT * FROM exampledb.examptable;'. Below the query is a results grid showing the data from the table.

	idexamptable	examptablecol
1	the	
2	quick	
3	brown	
4	fox	
5	jumped	
6	over	
7	the	
8	lazy	
9	dog	
*	HULL	HULL

For more information, see [Create a Table](#) in the MySQL documentation and the [MySQL Workbench product page](#).

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

Copy MySQL Data Using the Command Line

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

Prerequisites

Before you begin, you must complete the following steps:

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

Tasks

- [Define a Pipeline in JSON Format](#)
- [Upload and Activate the Pipeline Definition](#)

Define a Pipeline in JSON Format

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

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

 Note

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

```
{  
  "objects": [  
    {  
      "id": "ScheduleId113",  
      "startDateTime": "2013-08-26T00:00:00",  
      "name": "My Copy Schedule",  
      "type": "Schedule",  
      "period": "1 Days"  
    },  
    {  
      "id": "CopyActivityId112",  
      "input": {  
        "ref": " MySqlDataNodeId115"  
      },  
      "schedule": {  
        "ref": "ScheduleId113"  
      },  
      "name": "My Copy",  
      "runsOn": {  
        "ref": "Ec2ResourceId116"  
      },  
      "onSuccess": {  
        "ref": "ActionId1"  
      },  
      "onFail": {  
        "ref": "SnsAlarmId117"  
      },  
      "output": {  
        "ref": " S3DataNodeId114"  
      },  
      "type": "CopyActivity"  
    },  
    {  
      "id": "S3DataNodeId114",  
      "name": "My S3 Data Node",  
      "type": "DataNode",  
      "path": "  
    }  
  ]  
}
```

```
"schedule": {
    "ref": "ScheduleId113"
},
"filePath": "s3://amzn-s3-demo-bucket/rds-output/output.csv",
"name": "My S3 Data",
"type": "S3DataNode"
},
{
"id": "MySQLDataNodeId115",
"username": "my-username",
"schedule": {
    "ref": "ScheduleId113"
},
"name": "My RDS Data",
"*password": "my-password",
"table": "table-name",
"connectionString": "jdbc:mysql://your-sql-instance-name.id.region-name.rds.amazonaws.com:3306/database-name",
"selectQuery": "select * from #{table}",
"type": "SqlDataNode"
},
{
"id": "Ec2ResourceId116",
"schedule": {
    "ref": "ScheduleId113"
},
"name": "My EC2 Resource",
"role": "DataPipelineDefaultRole",
"type": "Ec2Resource",
"resourceRole": "DataPipelineDefaultResourceRole"
},
{
"message": "This is a success message.",
"id": "ActionId1",
"subject": "RDS to S3 copy succeeded!",
"name": "My Success Alarm",
"role": "DataPipelineDefaultRole",
"topicArn": "arn:aws:sns:us-east-1:123456789012:example-topic",
"type": "SnsAlarm"
},
{
"id": "Default",
"scheduleType": "timeseries",
"failureAndRerunMode": "CASCADE",
```

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

MySQL Data Node

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

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

Id

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

Username

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

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. The password field is preceded by the asterisk special character. For more information, see [Special Characters](#).

Table

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

connectionString

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

selectQuery

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

Type

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

Note

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

Amazon S3 Data Node

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

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

Id

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

Schedule

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

filePath

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

Name

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

Type

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

Resource

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

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

Id

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

Schedule

The schedule on which to create this computational resource.

Name

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

Role

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

Type

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

resourceRole

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

Activity

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

```
{  
  "id": "CopyActivityId112",  
  "input": {  
    "ref": "MySQLDataNodeId115"  
  },  
  "schedule": {  
    "ref": "ScheduleId113"  
  },  
  "name": "My Copy",  
  "runsOn": {  
    "ref": "Ec2ResourceId116"  
  },  
  "onSuccess": {  
    "ref": "ActionId1"  
  },  
  "onFail": {  
    "ref": "SnsAlarmId117"  
  },  
  "output": {  
    "ref": "S3DataNodeId114"  
  },  
  "type": "CopyActivity"  
},
```

Id

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

Input

The location of the MySQL data to copy

Schedule

The schedule on which to run this activity

Name

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

runsOn

The computational resource that performs the work that this activity defines. In this example, we provide a reference to the EC2 instance defined previously. Using the runsOn field causes AWS Data Pipeline to create the EC2 instance for you. The runsOn field indicates that the

resource exists in the AWS infrastructure, while the workerGroup value indicates that you want to use your own on-premises resources to perform the work.

onSuccess

The [SnsAlarm](#) to send if the activity completes successfully

onFail

The [SnsAlarm](#) to send if the activity fails

Output

The Amazon S3 location of the CSV output file

Type

The type of activity to perform.

Upload and Activate the Pipeline Definition

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

AWS CLI

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

```
aws datapipeline create-pipeline --name pipeline_name --unique-id token
{
    "pipelineId": "df-00627471S0VYZEXAMPLE"
}
```

To upload your pipeline definition, use the following [put-pipeline-definition](#) command.

```
aws datapipeline put-pipeline-definition --pipeline-id df-00627471S0VYZEXAMPLE --
pipeline-definition file://MyEmrPipelineDefinition.json
```

If your 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-00627471S0VYZEXAMPLE
```

You can verify that your pipeline appears in the pipeline list using the following [list-pipelines](#) command.

```
aws datapipeline list-pipelines
```

Copy Data to Amazon Redshift Using AWS Data Pipeline

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

Amazon S3 is a web service that enables you to store data in the cloud. For more information, see the [Amazon Simple Storage Service User Guide](#).

Amazon Redshift is a data warehouse service in the cloud. For more information, see the [Amazon Redshift Management Guide](#).

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

Contents

- [Before You Begin: Configure COPY Options and Load Data](#)
- [Set up Pipeline, Create a Security Group, and Create an Amazon Redshift Cluster](#)
- [Copy Data to Amazon Redshift Using the Command Line](#)

Before You Begin: Configure COPY Options and Load Data

Before copying data to Amazon Redshift within AWS Data Pipeline, ensure that you:

- Load data from Amazon S3.
- Set up the COPY activity in Amazon Redshift.

Once you have these options working and successfully complete a data load, transfer these options to AWS Data Pipeline, for performing the copying within it.

For COPY options, see [COPY](#) in the Amazon Redshift *Database Developer Guide*.

For steps to load data from Amazon S3, see [Loading data from Amazon S3](#) in the Amazon Redshift *Database Developer Guide*.

For example, the following SQL command in Amazon Redshift creates a new table named LISTING and copies sample data from a publicly available bucket in Amazon S3.

Replace the <iam-role-arn> and region with your own.

For details about this example, see [Load Sample Data from Amazon S3](#) in the Amazon Redshift *Getting Started Guide*.

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

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

Set up Pipeline, Create a Security Group, and Create an Amazon Redshift Cluster

To set up for the tutorial

1. Complete the tasks in [Setting up for AWS Data Pipeline](#).
2. Create a security group.
 - a. Open the Amazon EC2 console.
 - b. In the navigation pane, click **Security Groups**.
 - c. Click **Create Security Group**.
 - d. Specify a name and description for the security group.
 - e. [EC2-Classic] Select No VPC for **VPC**.

- f. [EC2-VPC] Select the ID of your VPC for **VPC**.
 - g. Click **Create**.
3. [EC2-Classic] Create an Amazon Redshift cluster security group and specify the Amazon EC2 security group.
- a. Open the Amazon Redshift console.
 - b. In the navigation pane, click **Security Groups**.
 - c. Click **Create Cluster Security Group**.
 - d. In the **Create Cluster Security Group** dialog box, specify a name and description for the cluster security group.
 - e. Click the name of the new cluster security group.
 - f. Click **Add Connection Type**.
 - g. In the **Add Connection Type** dialog box, select **EC2 Security Group** from **Connection Type**, select the security group that you created from **EC2 Security Group Name**, and then click **Authorize**.
4. [EC2-VPC] Create an Amazon Redshift cluster security group and specify the VPC security group.
- a. Open the Amazon EC2 console.
 - b. In the navigation pane, click **Security Groups**.
 - c. Click **Create Security Group**.
 - d. In the **Create Security Group** dialog box, specify a name and description for the security group, and select the ID of your VPC for **VPC**.
 - e. Click **Add Rule**. Specify the type, protocol, and port range, and start typing the ID of the security group in **Source**. Select the security group that you created in the second step.
 - f. Click **Create**.
5. The following is a summary of the steps.

If you have an existing Amazon Redshift cluster, make a note of the cluster ID.

To create a new cluster and load sample data, follow the steps in [Getting Started with Amazon Redshift](#). For more information about creating clusters, see [Creating a Cluster](#) in the *Amazon Redshift Management Guide*.

-
- a. Open the Amazon Redshift console.

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

Copy Data to Amazon Redshift Using the Command Line

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

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

Prerequisites

Before you begin, you must complete the following steps:

1. Install and configure a command line interface (CLI). For more information, see [Accessing AWS Data Pipeline](#).
2. Ensure that the IAM roles named **DataPipelineDefaultRole** and **DataPipelineDefaultResourceRole** exist. The AWS Data Pipeline console creates these roles for you automatically. If you haven't used the AWS Data Pipeline console at least once, then you must create these roles manually. For more information, see [IAM Roles for AWS Data Pipeline](#).
3. Set up the COPY command in Amazon Redshift, since you will need to have these same options working when you perform the copying within AWS Data Pipeline. For information, see [Before You Begin: Configure COPY Options and Load Data](#).
4. Set up an Amazon Redshift database. For more information, see [Set up Pipeline, Create a Security Group, and Create an Amazon Redshift Cluster](#).

Tasks

- [Define a Pipeline in JSON Format](#)

- [Upload and Activate the Pipeline Definition](#)

Define a Pipeline in JSON Format

This example scenario shows how to copy data from an Amazon S3 bucket to Amazon Redshift.

This is the full pipeline definition JSON file followed by an explanation for each of its sections. We recommend that you use a text editor that can help you verify the syntax of JSON-formatted files, and name the file using the .json file extension.

```
{  
  "objects": [  
    {  
      "id": "CSVId1",  
      "name": "DefaultCSV1",  
      "type": "CSV"  
    },  
    {  
      "id": "RedshiftDatabaseId1",  
      "databaseName": "dbname",  
      "username": "user",  
      "name": "DefaultRedshiftDatabase1",  
      "*password": "password",  
      "type": "RedshiftDatabase",  
      "clusterId": "redshiftclusterId"  
    },  
    {  
      "id": "Default",  
      "scheduleType": "timeseries",  
      "failureAndRerunMode": "CASCADE",  
      "name": "Default",  
      "role": "DataPipelineDefaultRole",  
      "resourceRole": "DataPipelineDefaultResourceRole"  
    },  
    {  
      "id": "RedshiftDataNodeId1",  
      "schedule": {  
        "ref": "ScheduleId1"  
      },  
      "tableName": "orders",  
      "name": "DefaultRedshiftDataNode1",  
      "type": "RedshiftDataNode"  
    }  
  ]  
}
```

```
"createTableSql": "create table StructuredLogs (requestBeginTime CHAR(30) PRIMARY KEY DISTKEY SORTKEY, requestEndTime CHAR(30), hostname CHAR(100), requestDate varchar(20));",
  "type": "RedshiftDataNode",
  "database": {
    "ref": "RedshiftDatabaseId1"
  },
},
{
  "id": "Ec2ResourceId1",
  "schedule": {
    "ref": "ScheduleId1"
  },
  "securityGroups": "MySecurityGroup",
  "name": "DefaultEc2Resource1",
  "role": "DataPipelineDefaultRole",
  "logUri": "s3://myLogs",
  "resourceRole": "DataPipelineDefaultResourceRole",
  "type": "Ec2Resource"
},
{
  "id": "ScheduleId1",
  "startDateTime": "yyyy-mm-ddT00:00:00",
  "name": "DefaultSchedule1",
  "type": "Schedule",
  "period": "period",
  "endDateTime": "yyyy-mm-ddT00:00:00"
},
{
  "id": "S3DataNodeId1",
  "schedule": {
    "ref": "ScheduleId1"
  },
  "filePath": "s3://datapipeline-us-east-1/samples/hive-ads-samples.csv",
  "name": "DefaultS3DataNode1",
  "dataFormat": {
    "ref": "CSVID1"
  },
  "type": "S3DataNode"
},
{
  "id": "RedshiftCopyActivityId1",
  "input": {
    "ref": "S3DataNodeId1"
```

```
        },
        "schedule": {
            "ref": "ScheduleId1"
        },
        "insertMode": "KEEP_EXISTING",
        "name": "DefaultRedshiftCopyActivity1",
        "runsOn": {
            "ref": "Ec2ResourceId1"
        },
        "type": "RedshiftCopyActivity",
        "output": {
            "ref": "RedshiftDataNodeId1"
        }
    }
]
}
```

For more information about these objects, see the following documentation.

Objects

- [Data Nodes](#)
- [Resource](#)
- [Activity](#)

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](#).

This input component is defined by the following fields:

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

```
"dataFormat": {  
    "ref": "CSVId1"  
},  
"type": "S3DataNode"  
},
```

id

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

schedule

A reference to the schedule component.

filePath

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

name

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

dataFormat

A reference to the format of the data for the activity to process.

Output Data Node

The output RedshiftDataNode pipeline component defines a location for the output data; in this case, a table in an Amazon Redshift database. For more information, see [RedshiftDataNode](#). This output component is defined by the following fields:

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

```
},  
},
```

id

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

schedule

A reference to the schedule component.

tableName

The name of the Amazon Redshift table.

name

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

createTableSql

A SQL expression to create the table in the database.

database

A reference to the Amazon Redshift database.

Database

The RedshiftDatabase component is defined by the following fields. For more information, see [RedshiftDatabase](#).

```
{  
  "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](#).

The Ec2Resource is defined by the following fields:

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

id

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

schedule

The schedule on which to create this computational resource.

securityGroups

The security group to use for the instances in the resource pool.

name

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

role

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

logUri

The Amazon S3 destination path to back up Task Runner logs from the Ec2Resource.

resourceRole

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

Activity

The last section in the JSON file is the definition of the activity that represents the work to perform. In this case, we use a RedshiftCopyActivity component to copy data from Amazon S3 to Amazon Redshift. For more information, see [RedshiftCopyActivity](#).

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-00627471S0VYZEXAMPLE"
}
```

To upload your pipeline definition, use the following [put-pipeline-definition](#) command.

```
aws datapipeline put-pipeline-definition --pipeline-id df-00627471S0VYZEXAMPLE --
pipeline-definition file://MyEmrPipelineDefinition.json
```

If your 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-00627471S0VYZEXAMPLE
```

You can verify that your pipeline appears in the pipeline list using the following [list-pipelines](#) command.

```
aws datapipeline list-pipelines
```

Pipeline Expressions and Functions

This section explains the syntax for using expressions and functions in pipelines, including the associated data types.

Simple Data Types

The following types of data can be set as field values.

Types

- [DateTime](#)
- [Numeric](#)
- [Object References](#)
- [Period](#)
- [String](#)

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](#). For more information about the pipeline object types, see [Pipeline Object Reference](#).

Period

Indicates how often a scheduled event should run. It's expressed in the format " N [years|months|weeks|days|hours|minutes]", where N is a positive integer value.

The minimum period is 15 minutes and the maximum period is 3 years.

The following example sets the period field of the Schedule object to 3 hours. This creates a schedule that runs every three hours.

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

String

Standard string values. Strings must be surrounded by double quotes (""). You can use the backslash character (\) to escape characters in a string. Multiline strings are not supported.

The following examples show examples of valid string values for the id field.

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

Strings can also contain expressions that evaluate to string values. These are inserted into the string, and are delimited with: "#{" and "}"". The following example uses an expression to insert the name of the current object into a path.

```
"filePath" : "s3://amzn-s3-demo-bucket/#${name}.csv"
```

For more information about using expressions, see [Referencing Fields and Objects](#) and [Expression Evaluation](#).

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](#).

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

Referencing Fields and Objects

Expressions can use fields of the current object where the expression exists, or fields of another object that is linked by a reference.

A slot format consists of a creation time followed by the object creation time, such as @S3BackupLocation_2018-01-31T11:05:33.

You can also reference the exact slot ID specified in the pipeline definition, such as the slot ID of the Amazon S3 backup location. To reference the slot ID, use #{{parent.@id}}.

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://amzn-s3-demo-bucket/ExampleDataNode.csv".

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

To use a field that exists on another object linked by a reference, use the node keyword. This keyword is only available with alarm and precondition objects.

Continuing with the previous example, an expression in an SnsAlarm can refer to the date and time range in a Schedule, because the S3DataNode references both.

Specifically, FailureNotify's message field can use the @scheduledStartTime and @scheduledEndTime runtime fields from ExampleSchedule, because ExampleDataNode's onFail field references FailureNotify and its schedule field references ExampleSchedule.

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

Note

You can create pipelines that have dependencies, such as tasks in your pipeline that depend on the work of other systems or tasks. If your pipeline requires certain resources, add those dependencies to the pipeline using preconditions that you associate with data nodes and tasks. This makes your pipelines easier to debug and more resilient. Additionally, keep your dependencies within a single pipeline when possible, because cross-pipeline troubleshooting is difficult.

Nested Expressions

AWS Data Pipeline allows you to nest values to create more complex expressions. For example, to perform a time calculation (subtract 30 minutes from the scheduledStartTime) and format the result to use in a pipeline definition, you could use the following expression in an activity:

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

and using the node prefix if the expression is part of an SnsAlarm or Precondition:

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

Lists

Expressions can be evaluated on lists and functions on lists. For example, assume that a list is defined like the following: "myList": ["one", "two"]. If this list is used in the expression

`#{'this is ' + myList}`, it will evaluate to `["this is one", "this is two"]`. If you have two lists, Data Pipeline will ultimately flatten them in their evaluation. For example, if `myList1` is defined as `[1,2]` and `myList2` is defined as `[3,4]` then the expression `#[{myList1}, #{myList2}]` will evaluate to `[1,2,3,4]`.

Node Expression

AWS Data Pipeline uses the `#{node.*}` expression in either `SnsAlarm` or `PreCondition` for a back-reference to a pipeline component's parent object. Since `SnsAlarm` and `PreCondition` are referenced from an activity or resource with no reference back from them, `node` provides the way to refer to the referrer. For example, the following pipeline definition demonstrates how a failure notification can use `node` to make a reference to its parent, in this case `ShellCommandActivity`, and include the parent's scheduled start and end times in the `SnsAlarm` message. The `scheduledStartTime` reference on `ShellCommandActivity` does not require the `node` prefix because `scheduledStartTime` refers to itself.

Note

The fields preceded by the AT (@) sign indicate those fields are runtime fields.

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

AWS Data Pipeline supports transitive references for user-defined fields, but not runtime fields. A transitive reference is a reference between two pipeline components that depends on another pipeline component as the intermediary. The following example shows a reference to a transitive user-defined field and a reference to a non-transitive runtime field, both of which are valid. For more information, see [User-defined fields](#).

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

Expression Evaluation

AWS Data Pipeline provides a set of functions that you can use to calculate the value of a field. The following example uses the `makeDate` function to set the `startDateTime` field of a `Schedule` object to "2011-05-24T0:00:00" GMT/UTC.

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

Mathematical Functions

The following functions are available for working with numerical values.

Function	Description
+	Addition.

Function	Description
+	Example: <code>#{1 + 2}</code> Result: 3
-	Subtraction. Example: <code>#{1 - 2}</code> Result: -1
*	Multiplication. Example: <code>#{1 * 2}</code> Result: 2
/	Division. If you divide two integers, the result is truncated. Example: <code>#{1 / 2}</code> , Result: 0 Example: <code>#{1.0 / 2}</code> , Result: .5
^	Exponent. Example: <code>#{2 ^ 2}</code> Result: 4.0

String Functions

The following functions are available for working with string values.

Function	Description
+	Concatenation. Non-string values are first converted to strings.

Function	Description
	Example: #{"hel" + "lo"} Result: "hello"

Date and Time Functions

The following functions are available for working with DateTime values. For the examples, the value of myDateTime is May 24, 2011 @ 5:10 pm GMT.

 **Note**

The date/time format for AWS Data Pipeline is Joda Time, which is a replacement for the Java date and time classes. For more information, see [Joda Time - Class DateTimeFormat](#).

Function	Description
<code>int day(DateTime myDateTime)</code>	Gets the day of the DateTime value as an integer. Example: #{{day(myDateTime)} Result: 24
<code>int dayOfYear(DateTime myDateTime)</code>	Gets the day of the year of the DateTime value as an integer. Example: #{{dayOfYear(myDateTime)} Result: 144
<code>DateTime firstOfMonth(DateTime myDateTime)</code>	Creates a DateTime object for the start of the month in the specified DateTime.

Function	Description
	<p>Example: <code>#{firstOfMonth(myDateTime)}</code></p> <p>Result: "2011-05-01T17:10:00z"</p>
<pre>String format(DateTime myDateTime, String format)</pre>	<p>Creates a String object that is the result of converting the specified DateTime using the specified format string.</p> <p>Example: <code>#{format(myDateTime, 'YYYY-MM-dd HH:mm:ss z')}</code></p> <p>Result: "2011-05-24T17:10:00 UTC"</p>
<pre>int hour(DateTime myDateTime)</pre>	<p>Gets the hour of the DateTime value as an integer.</p> <p>Example: <code>#{hour(myDateTime)}</code></p> <p>Result: 17</p>
<pre>DateTime makeDate(int year, int month, int day)</pre>	<p>Creates a DateTime object, in UTC, with the specified year, month, and day, at midnight.</p> <p>Example: <code>#{makeDate(2011, 5, 24)}</code></p> <p>Result: "2011-05-24T0:00:00z"</p>

Function	Description
<pre>DateTime makeDateTime(int year,int month,int day,int hour,int minute)</pre>	<p>Creates a DateTime object, in UTC, with the specified year, month, day, hour, and minute.</p> <p>Example: #{}{makeDateTime(2011,5,24,14,21)}</p> <p>Result: "2011-05-24T14:21:00z"</p>
<pre>DateTime midnight(DateTime myDateTime)</pre>	<p>Creates a DateTime object for the current midnight, relative to the specified DateTime. For example, where MyDateTime is 2011-05-25T17:10:00z , the result is as follows.</p> <p>Example: #{}{midnight(myDateTime)}</p> <p>Result: "2011-05-25T0:00:00z"</p>
<pre>DateTime minusDays(DateTime myDateTime,int daysToSub)</pre>	<p>Creates a DateTime object that is the result of subtracting the specified number of days from the specified DateTime.</p> <p>Example: #{}{minusDays(myDateTime,1)}</p> <p>Result: "2011-05-23T17:10:00z"</p>

Function	Description
<pre>DateTime minusHours(DateTime myDateTime,int hoursToSub)</pre>	<p>Creates a DateTime object that is the result of subtracting the specified number of hours from the specified DateTime.</p> <p>Example: <code>#{minusHours(myDateTime,1)}</code></p> <p>Result: "2011-05-24T16:10:00z"</p>
<pre>DateTime minusMinutes(DateTime myDateTim e,int minutesToSub)</pre>	<p>Creates a DateTime object that is the result of subtracting the specified number of minutes from the specified DateTime.</p> <p>Example: <code>#{minusMi nutes(myDateTime,1)}</code></p> <p>Result: "2011-05-24T17:09:00z"</p>
<pre>DateTime minusMonths(DateTime myDateTime,int monthsToSub)</pre>	<p>Creates a DateTime object that is the result of subtracting the specified number of months from the specified DateTime.</p> <p>Example: <code>#{minusMo nths(myDateTime,1)}</code></p> <p>Result: "2011-04-24T17:10:00z"</p>

Function	Description
<pre>DateTime minusWeeks(DateTime myDateTime,int weeksToSub)</pre>	<p>Creates a DateTime object that is the result of subtracting the specified number of weeks from the specified DateTime.</p> <p>Example: <code>#{minusWeeks(myDateTime,1)}</code></p> <p>Result: "2011-05-17T17:10:00z"</p>
<pre>DateTime minusYears(DateTime myDateTime,int yearsToSub)</pre>	<p>Creates a DateTime object that is the result of subtracting the specified number of years from the specified DateTime.</p> <p>Example: <code>#{minusYears(myDateTime,1)}</code></p> <p>Result: "2010-05-24T17:10:00z"</p>
<pre>int minute(DateTime myDateTime)</pre>	<p>Gets the minute of the DateTime value as an integer.</p> <p>Example: <code>#{minute(myDateTime)}</code></p> <p>Result: 10</p>

Function	Description
<pre>int month(DateTime myDateTime)</pre>	<p>Gets the month of the DateTime value as an integer.</p> <p>Example: #{month(myDateTime)}</p> <p>Result: 5</p>
<pre>DateTime plusDays(DateTime myDateTime, int daysToAdd)</pre>	<p>Creates a DateTime object that is the result of adding the specified number of days to the specified DateTime.</p> <p>Example: #{plusDays(myDateTime,1)}</p> <p>Result: "2011-05-25T17:10:00z"</p>
<pre>DateTime plusHours(DateTime myDateTime, int hoursToAdd)</pre>	<p>Creates a DateTime object that is the result of adding the specified number of hours to the specified DateTime.</p> <p>Example: #{plusHours(myDateTime,1)}</p> <p>Result: "2011-05-24T18:10:00z"</p>

Function	Description
<pre>DateTime plusMinutes(DateTime myDateTime,int minutesToAdd)</pre>	<p>Creates a DateTime object that is the result of adding the specified number of minutes to the specified DateTime.</p> <p>Example: <code>#{plusMinutes(myDateTime,1)}</code></p> <p>Result: "2011-05-24 17:11:00z"</p>
<pre>DateTime plusMonths(DateTime myDateTime,int monthsToAdd)</pre>	<p>Creates a DateTime object that is the result of adding the specified number of months to the specified DateTime.</p> <p>Example: <code>#{plusMonths(myDateTime,1)}</code></p> <p>Result: "2011-06-24T17:10:00z"</p>
<pre>DateTime plusWeeks(DateTime myDateTime,int weeksToAdd)</pre>	<p>Creates a DateTime object that is the result of adding the specified number of weeks to the specified DateTime.</p> <p>Example: <code>#{plusWeeks(myDateTime,1)}</code></p> <p>Result: "2011-05-31T17:10:00z"</p>

Function	Description
<pre>DateTime plusYears(DateTime myDateTime, int yearsToAdd)</pre>	<p>Creates a DateTime object that is the result of adding the specified number of years to the specified DateTime.</p> <p>Example: <code>#{plusYears(myDateTime,1)}</code></p> <p>Result: "2012-05-24T17:10:00z"</p>
<pre>DateTime sunday(DateTime myDateTime)</pre>	<p>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.</p> <p>Example: <code>#{sunday(myDateTime)}</code></p> <p>Result: "2011-05-22 17:10:00 UTC"</p>
<pre>int year(DateTime myDateTime)</pre>	<p>Gets the year of the DateTime value as an integer.</p> <p>Example: <code>#{year(myDateTime)}</code></p> <p>Result: 2011</p>

Function	Description
<code>DateTime yesterday(DateTime myDateTime)</code>	<p>Creates a DateTime object for the previous day, relative to the specified DateTime. The result is the same as <code>minusDays(1)</code>.</p> <p>Example: <code>#{yesterday(myDateTime)}</code></p> <p>Result: "2011-05-23T17:10:00z"</p>

Special Characters

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

Special Character	Description	Examples
@	Runtime field. This character is a field name prefix for a field that is only available when a pipeline runs.	<code>@actualStartTime</code> <code>@failureReason</code> <code>@resourceStatus</code>
#	Expression. Expressions are delimited by: "#{" and "}" and the contents of the braces are evaluated by AWS Data Pipeline. For more information, see Expressions .	<code>#{format(myDateTime,'YYYY-MM-dd hh:mm:ss')}</code> <code>s3://amzn-s3-demo-bucket/#{id}.csv</code>
*	Encrypted field. This character is a field name prefix to indicate that AWS Data Pipeline should encrypt	<code>*password</code>

Special Character	Description	Examples
	the contents of this field in transit between the console or CLI and the AWS Data Pipeline service.	

Pipeline Object Reference

You can use the following pipeline objects and components in your pipeline definition.

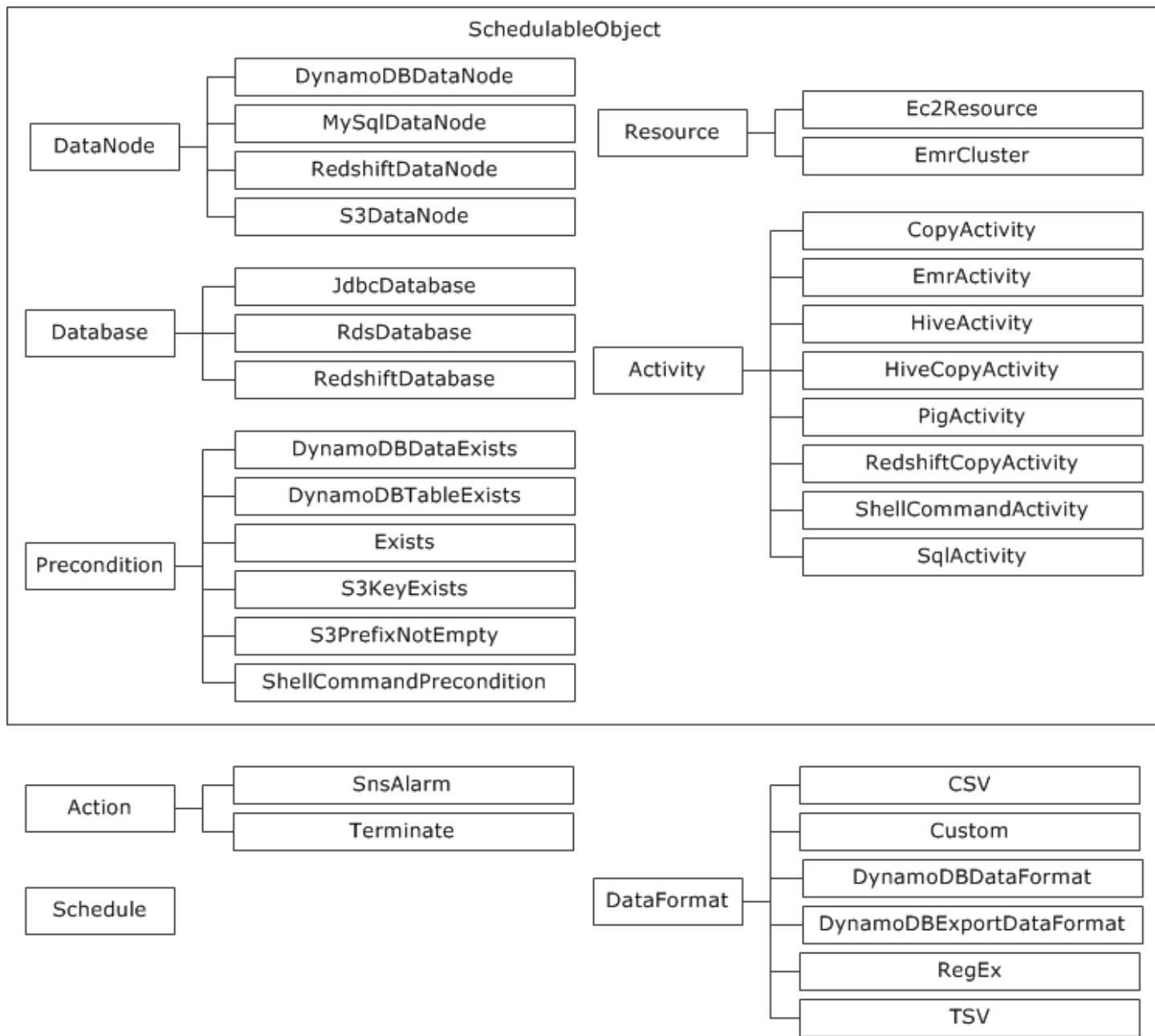
Contents

- [Data Nodes](#)
- [Activities](#)
- [Resources](#)
- [Preconditions](#)
- [Databases](#)
- [Data Formats](#)
- [Actions](#)
- [Schedule](#)
- [Utilities](#)

 **Note**

For an example application that uses the AWS Data Pipeline Java SDK, see [Data Pipeline DynamoDB Export Java Sample](#) on GitHub.

The following is the object hierarchy for AWS Data Pipeline.



Data Nodes

The following are the AWS Data Pipeline data node objects:

Objects

- [DynamoDBDataNode](#)
- [MySqlDataNode](#)
- [RedshiftDataNode](#)

- [S3DataNode](#)
- [SqlDataNode](#)

DynamoDBDataNode

Defines a data node using DynamoDB, which is specified as an input to a `HiveActivity` or `EMRACTIVITY` object.

 **Note**

The `DynamoDBDataNode` object does not support the `Exists` precondition.

Example

The following is an example of this object type. This object references two other objects that you'd define in the same pipeline definition file. `CopyPeriod` is a `Schedule` object and `Ready` is a precondition object.

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

Syntax

Required Fields	Description	Slot Type
tableName	The DynamoDB table.	String

Object Invocation Fields	Description	Slot Type
schedule	<p>This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), users can create a parent object that has a schedule reference. For more information about example optional schedule configurations, see Schedule.</p>	Reference Object, for example, "schedule": {"ref": "myScheduleId"}

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If this field is set, then a remote activity that does not complete within the set time of starting may be retried.	Period
dataFormat	DataFormat for the data described by this data node. Currently supported for HiveActivity and HiveCopyActivity.	Reference Object, "dataFormat": {"ref": "myDynamoDBDataFormatId"}

Optional Fields	Description	Slot Type
dependsOn	Specify dependency on another runnable object	Reference Object, e.g. "dependsOn":{"ref":"myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
lateAfterTimeout	The elapsed time after pipeline start within which the object must complete. It is triggered only when the schedule type is not set to ondemand.	Period
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail":{"ref":"myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction":{"ref":"myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess":{"ref":"myActionId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent":{"ref":"myBaseObjectId"}
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String

Optional Fields	Description	Slot Type
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, e.g. "precondition": {"ref":"myPreconditionId"}
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 0.1 and 1.0, inclusively.	Double
region	The code for the region where the DynamoDB table exists. For example, us-east-1. This is used by HiveActivity when it performs staging for DynamoDB tables in Hive.	Enumeration
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set, then remote activities that do not report progress for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period
runsOn	The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.	Reference Object, e.g. "runsOn": {"ref": "myResourceId"}

Optional Fields	Description	Slot Type
scheduleType	<p>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the <code>ActivatePipeline</code> operation for each subsequent run. Values are: cron, ondemand, and timeseries.</p>	Enumeration
workerGroup	<p>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.</p>	String
writeThroughputPercent	<p>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.</p>	Double

Runtime Fields	Description	Slot Type
@activeInstances	<p>List of the currently scheduled active instance objects.</p>	Reference Object, e.g. "activeInstances":

Runtime Fields	Description	Slot Type
		{"ref":"myRunnable ObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of dependency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnable ObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errord	The errord if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String

Runtime Fields	Description	Slot Type
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref": "myRunnableObject Id"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

MySqlDataNode

Defines a data node using MySQL.

Note

The MySqlDataNode type is deprecated. We recommend that you use [SqlDataNode](#) instead.

Example

The following is an example of this object type. This object references two other objects that you'd define in the same pipeline definition file. CopyPeriod is a Schedule object and Ready is a precondition object.

```
{  
  "id" : "Sql Table",  
  "type" : "MySqlDataNode",  
  "schedule" : { "ref" : "CopyPeriod" },  
  "table" : "adEvents",  
  "username": "user_name",  
  "*password": "my_password",  
  "connectionString": "jdbc:mysql://mysqlinstance-rds.example.us-  
east-1.rds.amazonaws.com:3306/database_name",
```

```

"selectQuery" : "select * from #{table} where eventTime >=
'#{@scheduledStartTime.format('YYYY-MM-dd HH:mm:ss')}' and eventTime <
'#{@scheduledEndTime.format('YYYY-MM-dd HH:mm:ss')}''",
"precondition" : { "ref" : "Ready" }
}

```

Syntax

Required Fields	Description	Slot Type
table	The name of the table in the MySQL database.	String

Object Invocation Fields	Description	Slot Type
schedule	This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), users can create a parent object that has a schedule reference . For more information about example optional schedule configurations, see https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html	Reference Object, e.g. "schedule": {"ref": "myScheduleId"}

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
createTableSql	An SQL create table expression that creates the table.	String
database	The name of the database.	Reference Object, e.g. "database": {"ref": "myDatabaseId"}
dependsOn	Specifies dependency on another runnable object.	Reference Object, e.g. "dependsOn": {"ref": "myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun.	Enumeration
insertQuery	An SQL statement to insert data into the table.	String
lateAfterTimeout	The elapsed time after pipeline start within which the object must complete. It is triggered only when the schedule type is not set to ondemand.	Period
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure	Integer

Optional Fields	Description	Slot Type
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref": "myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref": "myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref": "myActionId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref": "myBaseObjectId"}
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, e.g. "precondition": {"ref": "myPreconditionId"}
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set, then remote activities that do not report progress for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period
runsOn	The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.	Reference Object, e.g. "runsOn": {"ref": "myResourceId"}

Optional Fields	Description	Slot Type
scheduleType	<p>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the <code>ActivatePipeline</code> operation for each subsequent run. Values are: cron, ondemand, and timeseries.</p>	Enumeration
schemaName	The name of the schema holding the table	String
selectQuery	A SQL statement to fetch data from the table.	String
workerGroup	The worker group. This is used for routing tasks. If you provide a <code>runsOn</code> value and <code>workerGroup</code> exists, <code>workerGroup</code> is ignored.	String

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnable ObjectId"}

Runtime Fields	Description	Slot Type
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of the dependency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnable ObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errord	The errord if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime

Runtime Fields	Description	Slot Type
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object.	DateTime
@scheduledStartTime	Schedule start time for object.	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref": "myRunnableObject", "Id"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object.	String

System Fields	Description	Slot Type
@pipelineId	Id of the pipeline to which this object belongs to.	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects.	String

See Also

- [S3DataNode](#)

RedshiftDataNode

Defines a data node using Amazon Redshift. RedshiftDataNode represents the properties of the data inside a database, such as a data table, used by your pipeline.

Example

The following is an example of this object type.

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

Syntax

Required Fields	Description	Slot Type
database	The database on which the table resides.	Reference Object, e.g. "database":{ "ref":

Required Fields	Description	Slot Type
		"myRedshiftDatabaseId"}
tableName	The name of the Amazon Redshift table. The table is created if it doesn't already exist and you've provided createTableSql.	String

Object Invocation Fields	Description	Slot Type
schedule	<p>This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), users can create a parent object that has a schedule reference. For more information about example optional schedule configurations, see https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html</p>	Reference Object, e.g. "schedule":{"ref": "myScheduleId"}

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
createTableSql	An 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, numberOfRows integer sortKey). AWS Data Pipeline runs the script in the createTableSql field if the table, specified by tableName, does not exist in the schema, specified by the schemaName field. For example, if you specify schemaName as mySchema but do not include mySchema in the createTableSql field, the table is created in the wrong schema (by default, it would be created in PUBLIC). This occurs because AWS Data Pipeline does not parse your CREATE TABLE statements.	String
dependsOn	Specify dependency on another runnable object	Reference Object, e.g. "dependsOn": {"ref": "myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
lateAfterTimeout	The elapsed time after pipeline start within which the object must complete. It is triggered only when the schedule type is not set to ondemand.	Period

Optional Fields	Description	Slot Type
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	The maximum number attempt retries on failure.	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref": "myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref": "myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref": "myActionId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref": "myBaseObjectId"}
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, e.g. "precondition": {"ref": "myPreconditionId"}

Optional Fields	Description	Slot Type
primaryKeys	<p>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 an Amazon Redshift table, this setting overrides the existing key.</p>	String
reportProgressTime out	<p>Timeout for remote work successive calls to reportProgress. If set, then remote activities that do not report progress for the specified period may be considered stalled and so retried.</p>	Period
retryDelay	<p>The timeout duration between two retry attempts.</p>	Period
runsOn	<p>The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.</p>	Reference Object, e.g. "runsOn": {"ref": "myResourceId"}

Optional Fields	Description	Slot Type
scheduleType	<p>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the <code>ActivatePipeline</code> operation for each subsequent run. Values are: cron, ondemand, and timeseries.</p>	Enumeration
schemaName	<p>This optional field specifies the name of the schema for the Amazon Redshift table. If not specified, the schema name is PUBLIC, which is the default schema in Amazon Redshift. For more information, see the Amazon Redshift Database Developer Guide.</p>	String
workerGroup	<p>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.</p>	String

Runtime Fields	Description	Slot Type
@activeInstances	<p>List of the currently scheduled active instance objects.</p>	Reference Object, e.g. "activeInstances":

Runtime Fields	Description	Slot Type
		{"ref":"myRunnable ObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of the dependency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnable ObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String

Runtime Fields	Description	Slot Type
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref": "myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object.	String
@pipelineId	Id of the pipeline to which this object belongs to.	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects.	String

S3DataNode

Defines a data node using Amazon S3. By default, the S3DataNode uses server-side encryption. If you would like to disable this, set s3EncryptionType to NONE.

 **Note**

When you use an S3DataNode as input to CopyActivity, only the CSV and TSV data formats are supported.

Example

The following is an example of this object type. This object references another object that you'd define in the same pipeline definition file. CopyPeriod is a Schedule object.

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

Syntax

Object Invocation Fields	Description	Slot Type
schedule	<p>This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), users can create a parent object that has a schedule reference. For more information about example optional schedule configurations, see https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html</p>	Reference Object, e.g. "schedule": {"ref": "myScheduleId"}

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
compression	The type of compression for the data described by the S3DataNode. "none" is no compression and "gzip" is compressed with the	Enumeration

Optional Fields	Description	Slot Type
	gzip algorithm. This field is only supported for use with Amazon Redshift and when you use S3DataNode with CopyActivity.	
dataFormat	DataFormat for the data described by this S3DataNode.	Reference Object, e.g. "dataFormat": {"ref": "myDataFormatId"}
dependsOn	Specify dependency on another runnable object	Reference Object, e.g. "dependsOn": {"ref": "myActivityId"}
directoryPath	Amazon S3 directory path as a URI: s3://my-bucket/my-key-for-directory. You must provide either a filePath or directoryPath value.	String
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
filePath	The path to the object in Amazon S3 as a URI, for example: s3://my-bucket/my-key-for-file. You must provide either a filePath or directoryPath value. These represent a folder and a file name. Use the directoryPath value to accommodate multiple files in a directory.	String
lateAfterTimeout	The elapsed time after pipeline start within which the object must complete. It is triggered only when the schedule type is not set to ondemand.	Period
manifestFilePath	The Amazon S3 path to a manifest file in the format supported by Amazon Redshift. AWS Data Pipeline uses the manifest file to copy the specified Amazon S3 files into the table. This field is valid only when a RedShiftCopyActivity references the S3DataNode.	String

Optional Fields	Description	Slot Type
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref": "myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref": "myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref": "myActionId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref": "myBaseObjectId"}
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, e.g. "precondition": {"ref": "myPreconditionId"}
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set, then remote activities that do not report progress for the specified period may be considered stalled and so retried.	Period

Optional Fields	Description	Slot Type
retryDelay	The timeout duration between two retry attempts.	Period
runsOn	The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.	Reference Object, e.g. "runsOn": {"ref": "myResourceId"}
s3EncryptionType	Overrides the Amazon S3 encryption type. Values are SERVER_SIDE_ENCRYPTION or NONE. Server-side encryption is enabled by default.	Enumeration
scheduleType	<p>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the <code>ActivatePipeline</code> operation for each subsequent run. Values are: cron, ondemand, and timeseries.</p>	Enumeration
workerGroup	The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.	String

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnable ObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of the dependency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnable ObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String

Runtime Fields	Description	Slot Type
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref": "myRunnableObject Id"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

See Also

- [MySqlDataNode](#)

SqlDataNode

Defines a data node using SQL.

Example

The following is an example of this object type. This object references two other objects that you'd define in the same pipeline definition file. CopyPeriod is a Schedule object and Ready is a precondition object.

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

Syntax

Required Fields	Description	Slot Type
table	The name of the table in the SQL database.	String

Object Invocation Fields	Description	Slot Type
schedule	<p>This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), users can create a parent object that has a schedule reference. For more information about example optional schedule configurations, see https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html</p>	Reference Object, e.g. "schedule": {"ref": "myScheduleId"}

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String

Optional Fields	Description	Slot Type
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
createTableSql	An SQL create table expression that creates the table.	String
database	The name of the database.	Reference Object, e.g. "database": {"ref": "myDatabaseId"}
dependsOn	Specifies the dependency on another runnable object.	Reference Object, e.g. "dependsOn": {"ref": "myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun.	Enumeration
insertQuery	An SQL statement to insert data into the table.	String
lateAfterTimeout	The elapsed time after pipeline start within which the object must complete. It is triggered only when the schedule type is not set to ondemand.	Period
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref": "myActionId"}

Optional Fields	Description	Slot Type
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref": "myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref": "myActionId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref": "myBaseObjectId"}
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, e.g. "precondition": {"ref": "myPreconditionId"}
reportProgressTime out	Timeout for remote work successive calls to reportProgress. If set, then remote activities that do not report progress for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period
runsOn	The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.	Reference Object, e.g. "runsOn": {"ref": "myResourceId"}

Optional Fields	Description	Slot Type
scheduleType	<p>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the <code>ActivatePipeline</code> operation for each subsequent run. Values are: cron, ondemand, and timeseries.</p>	Enumeration
schemaName	The name of the schema holding the table	String
selectQuery	A SQL statement to fetch data from the table.	String
workerGroup	The worker group. This is used for routing tasks. If you provide a <code>runsOn</code> value and <code>workerGroup</code> exists, <code>workerGroup</code> is ignored.	String

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnable ObjectId"}

Runtime Fields	Description	Slot Type
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of the dependency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnable ObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errord	The errord if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime

Runtime Fields	Description	Slot Type
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref": "myRunnableObject", "Id"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String

System Fields	Description	Slot Type
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

See Also

- [S3DataNode](#)

Activities

The following are the AWS Data Pipeline activity objects:

Objects

- [CopyActivity](#)
- [EmrActivity](#)
- [HadoopActivity](#)
- [HiveActivity](#)
- [HiveCopyActivity](#)
- [PigActivity](#)
- [RedshiftCopyActivity](#)
- [ShellCommandActivity](#)
- [SqlActivity](#)

CopyActivity

Copies data from one location to another. CopyActivity supports [S3DataNode](#) and [SqlDataNode](#) as input and output and the copy operation is normally performed record-by-record. However,

CopyActivity provides a high-performance Amazon S3 to Amazon S3 copy when all the following conditions are met:

- The input and output are S3DataNodes
- The dataFormat field is the same for input and output

If you provide compressed data files as input and do not indicate this using the compression field on the S3 data nodes, CopyActivity might fail. In this case, CopyActivity does not properly detect the end of record character and the operation fails. Further, CopyActivity supports copying from a directory to another directory and copying a file to a directory, but record-by-record copy occurs when copying a directory to a file. Finally, CopyActivity does not support copying multipart Amazon S3 files.

CopyActivity has specific limitations to its CSV support. When you use an S3DataNode as input for CopyActivity, you can only use a Unix/Linux variant of the CSV data file format for the Amazon S3 input and output fields. The Unix/Linux variant requires the following:

- The separator must be the "," (comma) character.
- The records are not quoted.
- The default escape character is ASCII value 92 (backslash).
- The end of record identifier is ASCII value 10 (or "\n").

Windows-based systems typically use a different end-of-record character sequence: a carriage return and line feed together (ASCII value 13 and ASCII value 10). You must accommodate this difference using an additional mechanism, such as a pre-copy script to modify the input data, to ensure that CopyActivity can properly detect the end of a record; otherwise, the CopyActivity fails repeatedly.

When using CopyActivity to export from a PostgreSQL RDS object to a TSV data format, the default NULL character is \n.

Example

The following is an example of this object type. This object references three other objects that you would define in the same pipeline definition file. CopyPeriod is a Schedule object and InputData and OutputData are data node objects.

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

Syntax

Object Invocation Fields	Description	Slot Type
schedule	<p>This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), users can create a parent object that has a schedule reference. For more information about example optional schedule configurations, see https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html</p>	Reference Object, e.g. "schedule": {"ref": "myScheduleId"}

Required Group (One of the following is required)	Description	Slot Type
runsOn	The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.	Reference Object, e.g. "runsOn":{"ref":"myResourceId"}
workerGroup	The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.	String

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
dependsOn	Specify dependency on another runnable object.	Reference Object, e.g. "dependsOn":{"ref":"myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
input	The input data source.	Reference Object, e.g. "input":{"ref":"myDataNodeId"}
lateAfterTimeout	The elapsed time after pipeline start within which the object must complete. It is triggered	Period

Optional Fields	Description	Slot Type
	only when the schedule type is not set to ondemand.	
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail":{"ref":"myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction":{"ref":"myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess":{"ref":"myActionId"}
output	The output data source.	Reference Object, e.g. "output":{"ref":"myDataNodeId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent":{"ref":"myBaseObjectId"}
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, e.g. "precondition": {"ref":"myPreconditionId"}

Optional Fields	Description	Slot Type
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set, then remote activities that do not report progress for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period
scheduleType	Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the ActivatePipeline operation for each subsequent run. Values are: cron, ondemand, and timeseries.	Enumeration

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances":

Runtime Fields	Description	Slot Type
		{"ref":"myRunnable ObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of dependency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnable ObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String

Runtime Fields	Description	Slot Type
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref": "myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

See Also

- [ShellCommandActivity](#)
- [EmrActivity](#)
- [Export MySQL Data to Amazon S3 Using AWS Data Pipeline](#)

EmrActivity

Runs an EMR cluster.

AWS Data Pipeline uses a different format for steps than Amazon EMR; for example, AWS Data Pipeline uses comma-separated arguments after the JAR name in the EmrActivity step field. The following example shows a step formatted for Amazon EMR, followed by its AWS Data Pipeline equivalent:

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

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

Examples

The following is an example of this object type. This example uses older versions of Amazon EMR. Verify this example for correctness with the version of Amazon EMR cluster that you are using.

This object references three other objects that you would define in the same pipeline definition file. `MyEmrCluster` is an `EmrCluster` object and `MyS3Input` and `MyS3Output` are `S3DataNode` objects.

Note

In this example, you can replace the `step` field with your desired cluster string, which could be a Pig script, Hadoop streaming cluster, your own custom JAR including its parameters, or so on.

Hadoop 2.x (AMI 3.x)

```
{  
    "id" : "MyEmrActivity",  
    "type" : "EmrActivity",  
    "runsOn" : { "ref" : "MyEmrCluster" },  
    "preStepCommand" : "scp remoteFiles localFiles",  
    "step" : ["s3://amzn-s3-demo-bucket/myPath/myStep.jar,firstArg,secondArg,-files,s3://  
    amzn-s3-demo-bucket/myPath/myFile.py,-input,s3://myinputbucket/path,-output,s3://  
    myoutputbucket/path,-mapper,myFile.py,-reducer,reducerName","s3://amzn-s3-demo-bucket/  
    myPath/myotherStep.jar,..."],  
    "postStepCommand" : "scp localFiles remoteFiles",  
    "input" : { "ref" : "MyS3Input" },  
    "output" : { "ref" : "MyS3Output" }  
}
```

Note

To pass arguments to an application in a step, you need to specify the Region in the path of the script, as in the following example. In addition, you may need to escape the arguments that you pass. For example, if you use `script-runner.jar` to run a shell script and want to pass arguments to the script, you must escape the commas that separate them. The following step slot illustrates how to do this:

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

This step uses `script-runner.jar` to run the `echo.sh` shell script and passes `a`, `b`, and `c` as a single argument to the script. The first escape character is removed from the

resultant argument so you may need to escape again. For example, if you had `File\.gz` as an argument in JSON, you could escape it using `File\\\\.gz`. However, because the first escape is discarded, you must use `File\\\\\\\\\\\\.gz`.

Syntax

Object Invocation Fields	Description	Slot Type
schedule	<p>This object is invoked within the execution of a schedule interval. Specify a schedule reference to another object to set the dependency execution order for this object. You can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"} . In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), you can create a parent object that has a schedule reference. For more information about example optional schedule configurations, see https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html</p>	Reference Object, for example, "schedule": {"ref": "myScheduleId"}

Required Group (One of the following is required)	Description	Slot Type
runsOn	The Amazon EMR cluster on which this job will run.	Reference Object, for example, "runsOn": {"ref":"myEmrClusterId"}
workerGroup	The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.	String

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set, then a remote activity that does not complete within the set time of starting may be retried.	Period
dependsOn	Specify dependency on another runnable object.	Reference Object, for example, "dependsOn": {"ref":"myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun.	Enumeration
input	The location of the input data.	Reference Object, for example, "input": {"ref":"myDataNodeId"}

Optional Fields	Description	Slot Type
lateAfterTimeout	The elapsed time after pipeline start within which the object must complete. It is triggered only when the schedule type is not set to ondemand.	Period
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	The maximum number of attempt retries on failure.	Integer
onFail	An action to run when current object fails.	Reference Object, for example, "onFail": {"ref":"myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, for example, "onLateAction": {"ref":"myActionId"}
onSuccess	An action to run when the current object succeeds.	Reference Object, for example, "onSuccess": {"ref":"myActionId"}
output	The location of the output data.	Reference Object, for example, "output": {"ref":"myDataNode Id"}
parent	The parent of the current object from which slots will be inherited.	Reference Object, for example, "parent": {"ref":"myBaseObjectId"}

Optional Fields	Description	Slot Type
pipelineLogUri	The Amazon S3 URI, such as 's3://BucketName/Prefix/' for uploading logs for the pipeline.	String
postStepCommand	Shell scripts to be run after all steps are finished. To specify multiple scripts, up to 255, add multiple postStepCommand fields.	String
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, for example, "precondition": {"ref": "myPreconditionId"}
preStepCommand	Shell scripts to be run before any steps are run. To specify multiple scripts, up to 255, add multiple preStepCommand fields.	String
reportProgressTimeout	The timeout for remote work successive calls to reportProgress . If set, then remote activities that do not report progress for the specified period may be considered stalled and so retried.	Period

Optional Fields	Description	Slot Type
resizeClusterBeforeRunning	Resize the cluster before performing this activity to accommodate DynamoDB tables specified as inputs or outputs.	Boolean
	<p>Note</p> <p>If your <code>EmrActivity</code> uses a <code>DynamoDBDataNode</code> as either an input or output data node, and if you set the <code>resizeClusterBeforeRunning</code> to TRUE, AWS Data Pipeline starts using <code>m3.xlarge</code> instance types. This overwrites your instance type choices with <code>m3.xlarge</code>, which could increase your monthly costs.</p>	
resizeClusterMaxInstances	A limit on the maximum number of instances that can be requested by the resize algorithm.	Integer
retryDelay	The timeout duration between two retry attempts.	Period

Optional Fields	Description	Slot Type
scheduleType	<p>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of the interval, or end of the interval. Values are: <code>cron</code>, <code>ondemand</code>, and <code>timeseries</code> . The <code>timeseries</code> scheduling means that instances are scheduled at the end of each interval. The <code>cron</code> scheduling means that instances are scheduled at the beginning of each interval. An <code>ondemand</code> schedule allows you to run a pipeline one time per activation. You do not have to clone or re-create the pipeline to run it again. If you use an <code>ondemand</code> schedule, it must be specified in the default object and must be the only <code>scheduleType</code> specified for objects in the pipeline. To use <code>ondemand</code> pipelines, call the <code>ActivatePipeline</code> operation for each subsequent run.</p>	Enumeration
step	<p>One or more steps for the cluster to run. To specify multiple steps, up to 255, add multiple <code>step</code> fields. Use comma-separated arguments after the JAR name; for example, "s3://amzn-s3-demo-bucket/MyWork.jar, arg1,arg2,arg3 ".</p>	String

Runtime Fields	Description	Slot Type
@activeInstances	<p>List of the currently scheduled active instance objects.</p>	Reference Object, e.g. "activeInstances": {"ref":"myRunnable ObjectId"}

Runtime Fields	Description	Slot Type
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of the dependency chain the object failed on.	Reference Object, for example, "cascadeFailedOn": {"ref": "myRunnableObjectId"}
emrStepLog	Amazon EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime

Runtime Fields	Description	Slot Type
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for the object.	DateTime
@scheduledStartTime	Schedule start time for the object.	DateTime
@status	The status of this object.	String
@version	Pipeline version that the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, for example, "waitingOn":{"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object.	String

System Fields	Description	Slot Type
@pipelineId	ID of the pipeline to which this object belongs.	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects.	String

See Also

- [ShellCommandActivity](#)
- [CopyActivity](#)
- [EmrCluster](#)

HadoopActivity

Runs a MapReduce job on a cluster. The cluster can be an EMR cluster managed by AWS Data Pipeline or another resource if you use TaskRunner. Use HadoopActivity when you want to run work in parallel. This allows you to use the scheduling resources of the YARN framework or the MapReduce resource negotiator in Hadoop 1. If you would like to run work sequentially using the Amazon EMR Step action, you can still use [EmrActivity](#).

Examples

HadoopActivity using an EMR cluster managed by AWS Data Pipeline

The following HadoopActivity object uses an EmrCluster resource to run a program:

```
{  
  "name": "MyHadoopActivity",  
  "schedule": {"ref": "ResourcePeriod"},  
  "runsOn": {"ref": "MyEmrCluster"},  
  "type": "HadoopActivity",  
  "preActivityTaskConfig": {"ref": "preTaskScriptConfig"},  
  "jarUri": "/home/hadoop/contrib/streaming/hadoop-streaming.jar",  
  "argument": [  
    "-files",  
    "
```

```
"s3://elasticmapreduce/samples/wordcount/wordSplitter.py",
"-mapper",
"wordSplitter.py",
"-reducer",
"aggregate",
"-input",
"s3://elasticmapreduce/samples/wordcount/input/",
"-output",
"s3://amzn-s3-demo-bucket/MyHadoopActivity/#{@pipelineId}/
#{format(@scheduledStartTime, 'YYYY-MM-dd')}"

],
"maximumRetries": "0",
"postActivityTaskConfig": {"ref": "postTaskScriptConfig"},
"hadoopQueue" : "high"
}
```

Here is the corresponding *MyEmrCluster*, which configures the FairScheduler and queues in YARN for Hadoop 2-based AMIs:

```
{
  "id" : "MyEmrCluster",
  "type" : "EmrCluster",
  "hadoopSchedulerType" : "PARALLEL_FAIR_SCHEDULING",
  "amiVersion" : "3.7.0",
  "bootstrapAction" : ["s3://Region.elasticmapreduce/bootstrap-
actions/configure-hadoop,-z,yarn.scheduler.capacity.root.queues=low
\\,high\\,default,-z,yarn.scheduler.capacity.root.high.capacity=50,-
z,yarn.scheduler.capacity.root.low.capacity=10,-
z,yarn.scheduler.capacity.root.default.capacity=30"]
}
```

This is the EmrCluster you use to configure FairScheduler in Hadoop 1:

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

The following EmrCluster configures CapacityScheduler for Hadoop 2-based AMIs:

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

HadoopActivity using an existing EMR cluster

In this example, you use workergroups and a TaskRunner to run a program on an existing EMR cluster. The following pipeline definition uses HadoopActivity to:

- Run a MapReduce program only on *myWorkerGroup* resources. For more information about worker groups, see [Executing Work on Existing Resources Using Task Runner](#).
- Run a preActivityTaskConfig and postActivityTaskConfig

```
{  
    "objects": [  
        {  
            "argument": [  
                "-files",  
                "s3://elasticmapreduce/samples/wordcount/wordSplitter.py",  
                "-mapper",  
                "wordSplitter.py",  
                "-reducer",  
                "aggregate",  
                "-input",  
                "s3://elasticmapreduce/samples/wordcount/input/",  
                "-output",  
                "s3://amzn-s3-demo-bucket/MyHadoopActivity/#{@pipelineId}/  
                #{format(@scheduledStartTime, 'YYYY-MM-dd')}"  
            ],  
            "id": "MyHadoopActivity",  
            "jarUri": "/home/hadoop/contrib/streaming/hadoop-streaming.jar",  
            "name": "MyHadoopActivity",  
            "workerGroup": "myWorkerGroup",  
            "taskRunner": "mapreduce",  
            "preActivityTaskConfig": "s3://Region.elasticmapreduce/bootstrap-  
            actions/configure-hadoop,-z,yarn.scheduler.capacity.root.queues=low  
            \\\\\\,high,-z,yarn.scheduler.capacity.root.high.capacity=40,-  
            z,yarn.scheduler.capacity.root.low.capacity=60",  
            "postActivityTaskConfig": "s3://Region.elasticmapreduce/bootstrap-  
            actions/configure-hadoop,-z,yarn.scheduler.capacity.root.queues=low  
            \\\\\\,high,-z,yarn.scheduler.capacity.root.high.capacity=40,-  
            z,yarn.scheduler.capacity.root.low.capacity=60"  
        }  
    ]  
}
```

```
"type": "HadoopActivity"
},
{
  "id": "SchedulePeriod",
  "startDateTime": "start_datetime",
  "name": "SchedulePeriod",
  "period": "1 day",
  "type": "Schedule",
  "endDateTime": "end_datetime"
},
{
  "id": "ShellScriptConfig",
  "scriptUri": "s3://amzn-s3-demo-bucket/scripts/preTaskScript.sh",
  "name": "preTaskScriptConfig",
  "scriptArgument": [
    "test",
    "argument"
  ],
  "type": "ShellScriptConfig"
},
{
  "id": "ShellScriptConfig",
  "scriptUri": "s3://amzn-s3-demo-bucket/scripts/postTaskScript.sh",
  "name": "postTaskScriptConfig",
  "scriptArgument": [
    "test",
    "argument"
  ],
  "type": "ShellScriptConfig"
},
{
  "id": "Default",
  "scheduleType": "cron",
  "schedule": {
    "ref": "SchedulePeriod"
  },
  "name": "Default",
  "pipelineLogUri": "s3://amzn-s3-demo-bucket/
logs/2015-05-22T18:02:00.343Z642f3fe415",
  "maximumRetries": "0",
  "workerGroup": "myWorkerGroup",
  "preActivityTaskConfig": {
    "ref": "preTaskScriptConfig"
  }
},
```

```

    "postActivityTaskConfig": [
        {
            "ref": "postTaskScriptConfig"
        }
    ]
}
]
}

```

Syntax

Required Fields	Description	Slot Type
jarUri	Location of a JAR in Amazon S3 or the local file system of the cluster to run with HadoopActivity.	String

Object Invocation Fields	Description	Slot Type
schedule	This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), users can create a parent object that has a schedule reference . For more information about example optional schedule configurations, see https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html	Reference Object, e.g. "schedule": {"ref": "myScheduleId"}

Required Group (One of the following is required)	Description	Slot Type
runsOn	EMR Cluster on which this job will run.	Reference Object, e.g. "runsOn":{"ref":"myEmrClusterId"}
workerGroup	The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.	String

Optional Fields	Description	Slot Type
argument	Arguments to pass to the JAR.	String
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
dependsOn	Specify dependency on another runnable object.	Reference Object, e.g. "dependsOn":{"ref":"myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
hadoopQueue	The Hadoop scheduler queue name on which the activity will be submitted.	String

Optional Fields	Description	Slot Type
input	Location of the input data.	Reference Object, e.g. "input": {"ref": "myDataNodeId"}
lateAfterTimeout	The elapsed time after pipeline start within which the object must complete. It is triggered only when the schedule type is not set to ondemand.	Period
mainClass	The main class of the JAR you are executing with HadoopActivity.	String
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref": "myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref": "myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref": "myActionId"}
output	Location of the output data.	Reference Object, e.g. "output": {"ref": "myDataNodeId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref": "myBaseObjectId"}

Optional Fields	Description	Slot Type
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
postActivityTaskConfig	Post-activity configuration script to be run. This consists of a URI of the shell script in Amazon S3 and a list of arguments.	Reference Object, e.g. "postActivityTaskConfig": {"ref": "myShellScriptConfigId"}
preActivityTaskConfig	Pre-activity configuration script to be run. This consists of a URI of the shell script in Amazon S3 and a list of arguments.	Reference Object, e.g. "preActivityTaskConfig": {"ref": "myShellScriptConfigId"}
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, e.g. "precondition": {"ref": "myPreconditionId"}
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set, then remote activities that do not report progress for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period

Optional Fields	Description	Slot Type
scheduleType	<p>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the ActivatePipeline operation for each subsequent run. Values are: cron, ondemand, and timeseries.</p>	Enumeration

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnable ObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime

Runtime Fields	Description	Slot Type
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of the dependency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnable ObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errord	The errord if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime

Runtime Fields	Description	Slot Type
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref" : "myRunnableObject Id"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object.	String
@pipelineId	Id of the pipeline to which this object belongs to.	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects.	String

See Also

- [ShellCommandActivity](#)
- [CopyActivity](#)
- [EmrCluster](#)

HiveActivity

Runs a Hive query on an EMR cluster. `HiveActivity` makes it easier to set up an Amazon EMR activity and automatically creates Hive tables based on input data coming in from either Amazon S3 or Amazon RDS. All you need to specify is the HiveQL to run on the source data. AWS Data Pipeline automatically creates Hive tables with \${input1}, \${input2}, and so on, based on the input fields in the `HiveActivity` object.

For Amazon S3 inputs, the `dataFormat` field is used to create the Hive column names.

For MySQL (Amazon RDS) inputs, the column names for the SQL query are used to create the Hive column names.

 **Note**

This activity uses the Hive [CSV Serde](#).

Example

The following is an example of this object type. This object references three other objects that you define in the same pipeline definition file. `MySchedule` is a `Schedule` object and `MyS3Input` and `MyS3Output` are data node objects.

```
{  
  "name" : "ProcessLogData",  
  "id" : "MyHiveActivity",  
  "type" : "HiveActivity",  
  "schedule" : { "ref": "MySchedule" },  
  "hiveScript" : "INSERT OVERWRITE TABLE ${output1} select  
host,user,time,request,status,size from ${input1};",  
  "input" : { "ref": "MyS3Input" },  
  "output" : { "ref": "MyS3Output" },  
  "runsOn" : { "ref": "MyEmrCluster" }  
}
```

```
}
```

Syntax

Object Invocation Fields	Description	Slot Type
schedule	<p>This object is invoked within the execution of a schedule interval. Specify a schedule reference to another object to set the dependency execution order for this object. You can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "Defaultschedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), you can create a parent object that has a schedule reference. For more information about example optional schedule configurations, see https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html.</p>	Reference Object, e.g. "schedule": {"ref": "myScheduleId"}

Required Group (One of the following is required)	Description	Slot Type
hiveScript	The Hive script to run.	String
scriptUri	The location of the Hive script to run (for example, s3://scriptLocation).	String

Required Group	Description	Slot Type
runsOn	The EMR cluster on which this HiveActivity runs.	Reference Object, e.g. "runsOn": {"ref": "myEmrClusterId"}
workerGroup	The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.	String
input	The input data source.	Reference Object, such as "input": {"ref": "myDataNode1d"}
output	The output data source.	Reference Object, such as "output": {"ref": "myDataNodeId"}

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set, then a remote activity that does not complete within the set time of starting may be retried.	Period
dependsOn	Specify dependency on another runnable object.	Reference Object, such as "dependsOn": {"ref": "myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun.	Enumeration

Optional Fields	Description	Slot Type
hadoopQueue	The Hadoop scheduler queue name on which the job will be submitted.	String
lateAfterTimeout	The elapsed time after pipeline start within which the object must complete. It is triggered only when the schedule type is not set to ondemand.	Period
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	The maximum number of attempt retries on failure.	Integer
onFail	An action to run when current object fails.	Reference Object, such as "onFail": {"ref":"myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, such as "onLateAction": {"ref":"myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, such as "onSuccess": {"ref":"myActionId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, such as "parent": {"ref":"myBaseObjectID"}
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String

Optional Fields	Description	Slot Type
postActivityTaskConfig	Post-activity configuration script to be run. This consists of a URI of the shell script in Amazon S3 and a list of arguments.	Reference Object, such as "postActivityTaskConfig": {"ref": "myShellScriptConfigId"}
preActivityTaskConfig	Pre-activity configuration script to be run. This consists of a URI of the shell script in Amazon S3 and a list of arguments.	Reference Object, such as "preActivityTaskConfig": {"ref": "myShellScriptConfigId"}
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, such as "precondition": {"ref": "myPreconditionId"}
reportProgressTimeout	Timeout for remote work successive calls to <code>reportProgress</code> . If set, then remote activities that do not report progress for the specified period may be considered stalled and so retried.	Period

Optional Fields	Description	Slot Type
resizeClusterBeforeRunning	<p>Resize the cluster before performing this activity to accommodate DynamoDB data nodes specified as inputs or outputs.</p> <div data-bbox="502 439 638 475" style="border: 1px solid #ccc; border-radius: 50%; padding: 2px 5px; margin-bottom: 5px;"> Note</div> <p>If your activity uses a <code>DynamoDBDataNode</code> as either an input or output data node, and if you set the <code>resizeClusterBeforeRunning</code> to TRUE, AWS Data Pipeline starts using <code>m3.xlarge</code> instance types. This overwrites your instance type choices with <code>m3.xlarge</code>, which could increase your monthly costs.</p>	Boolean
resizeClusterMaxInstances	A limit on the maximum number of instances that can be requested by the resize algorithm.	Integer
retryDelay	The timeout duration between two retry attempts.	Period

Optional Fields	Description	Slot Type
scheduleType	<p>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the <code>ActivatePipeline</code> operation for each subsequent run. Values are: cron, ondemand, and timeseries.</p>	Enumeration
scriptVariable	<p>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: <code>SAMPLE=s3://elasticsearchmapreduce/samples/hive-ads</code> and <code>FILTER_DATE=#{format(@scheduledStartTime, 'YYYY-MM-dd')}%</code>. This field accepts multiple values and works with both <code>script</code> and <code>scriptUri</code> fields. In addition, <code>scriptVariable</code> functions regardless of whether stage is set to true or false. This field is especially useful to send dynamic values to Hive using AWS Data Pipeline expressions and functions.</p>	String

Optional Fields	Description	Slot Type
stage	Determines whether staging is enabled before or after running the script. Not permitted with Hive 11, so use an Amazon EMR AMI version 3.2.0 or greater.	Boolean

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, such as "activeInstances": {"ref": "myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of the dependency chain the object failed on.	Reference Object, such as "cascadeFailedOn": {"ref": "myRunnableObjectId"}
emrStepLog	Amazon EMR step logs available only on EMR activity attempts.	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String

Runtime Fields	Description	Slot Type
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for an object.	DateTime
@scheduledStartTime	Schedule start time for an object.	DateTime

Runtime Fields	Description	Slot Type
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, such as "waitingOn":{"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object.	String
@pipelineId	Id of the pipeline to which this object belongs.	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects.	String

See Also

- [ShellCommandActivity](#)
- [EmrActivity](#)

HiveCopyActivity

Runs a Hive query on an EMR cluster. HiveCopyActivity makes it easier to copy data between DynamoDB tables. HiveCopyActivity accepts a HiveQL statement to filter input data from DynamoDB at the column and row level.

Example

The following example shows how to use `HiveCopyActivity` and `DynamoDBExportDataFormat` to copy data from one `DynamoDBDataNode` to another, while filtering data, based on a time stamp.

```
{  
  "objects": [  
    {  
      "id" : "DataFormat.1",  
      "name" : "DataFormat.1",  
      "type" : "DynamoDBExportDataFormat",  
      "column" : "timeStamp BIGINT"  
    },  
    {  
      "id" : "DataFormat.2",  
      "name" : "DataFormat.2",  
      "type" : "DynamoDBExportDataFormat"  
    },  
    {  
      "id" : "DynamoDBDataNode.1",  
      "name" : "DynamoDBDataNode.1",  
      "type" : "DynamoDBDataNode",  
      "tableName" : "item_mapped_table_restore_temp",  
      "schedule" : { "ref" : "ResourcePeriod" },  
      "dataFormat" : { "ref" : "DataFormat.1" }  
    },  
    {  
      "id" : "DynamoDBDataNode.2",  
      "name" : "DynamoDBDataNode.2",  
      "type" : "DynamoDBDataNode",  
      "tableName" : "restore_table",  
      "region" : "us_west_1",  
      "schedule" : { "ref" : "ResourcePeriod" },  
      "dataFormat" : { "ref" : "DataFormat.2" }  
    },  
    {  
      "id" : "EmrCluster.1",  
      "name" : "EmrCluster.1",  
      "type" : "EmrCluster",  
      "schedule" : { "ref" : "ResourcePeriod" },  
      "masterInstanceType" : "m1.xlarge",  
      "coreInstanceCount" : "4"  
    }  
  ]  
}
```

```

},
{
  "id" : "HiveTransform.1",
  "name" : "Hive Copy Transform.1",
  "type" : "HiveCopyActivity",
  "input" : { "ref" : "DynamoDBDataNode.1" },
  "output" : { "ref" : "DynamoDBDataNode.2" },
  "schedule" :{ "ref" : "ResourcePeriod" },
  "runsOn" : { "ref" : "EmrCluster.1" },
  "filterSql" : "`timeStamp` > unix_timestamp(\"#{@scheduledStartTime}\", \"yyyy-MM-dd'T'HH:mm:ss\")"
},
{
  "id" : "ResourcePeriod",
  "name" : "ResourcePeriod",
  "type" : "Schedule",
  "period" : "1 Hour",
  "startDateTime" : "2013-06-04T00:00:00",
  "endDateTime" : "2013-06-04T01:00:00"
}
]
}

```

Syntax

Object Invocation Fields	Description	Slot Type
schedule	<p>This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within</p>	Reference Object, e.g. "schedule": {"ref": "myScheduleId"}

Object Invocation Fields	Description	Slot Type
	<p>the master schedule), users can create a parent object that has a schedule reference . For more information about example optional schedule configurations, see https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html</p>	

Required Group (One of the following is required)	Description	Slot Type
runsOn	Specify cluster to run on.	Reference Object, e.g. "runsOn": {"ref": "myResourceId"}
workerGroup	The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.	String

Optional Fields	Description	Slot Type
attemptStatus	The most recently reported status from the remote activity.	String
attemptTimeout	The timeout for remote work completion. If set, then a remote activity that does not complete within the set time of starting may be retried.	Period

Optional Fields	Description	Slot Type
dependsOn	Specifies the dependency on another runnable object.	Reference Object, e.g. "dependsOn":{"ref":"myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun.	Enumeration
filterSql	A Hive SQL statement fragment that filters a subset of DynamoDB or Amazon S3 data to copy. The filter should only contain predicates and not begin with a WHERE clause, because AWS Data Pipeline adds it automatically.	String
input	The input data source. This must be a S3DataNode or DynamoDBDataNode . If you use DynamoDBNode , specify a DynamoDBExportDataFormat .	Reference Object, e.g. "input":{"ref":"myDataNode1d"}
lateAfterTimeout	The elapsed time after pipeline start within which the object must complete. It is triggered only when the schedule type is not set to ondemand.	Period
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	The maximum number attempt retries on failure.	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail":{"ref":"myActionId"}

Optional Fields	Description	Slot Type
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref": "myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref": "myActionId"}
output	The output data source. If input is S3DataNode , this must be DynamoDBDataNode . Otherwise, this can be S3DataNode or DynamoDBDataNode . If you use DynamoDBNode , specify a DynamoDBExportDataFormat .	Reference Object, e.g. "output": {"ref": "myDataNodeId"}
parent	The parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref": "myBaseObjectId"}
pipelineLogUri	The Amazon S3 URI, such as 's3://BucketName/Key/' , for uploading logs for the pipeline.	String
postActivityTaskConfig	The post-activity configuration script to be run. This consists of a URI of the shell script in Amazon S3 and a list of arguments.	Reference Object, e.g. "postActivityTaskConfig": {"ref": "myShellScriptConfigId"}
preActivityTaskConfig	The pre-activity configuration script to be run. This consists of a URI of the shell script in Amazon S3 and a list of arguments.	Reference Object, e.g. "preActivityTaskConfig": {"ref": "myShellScriptConfigId"}

Optional Fields	Description	Slot Type
precondition	Optionally defines a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, e.g. "precondition": {"ref":"myPreconditionId"}
reportProgressTimeout	The timeout for remote work successive calls to <code>reportProgress</code> . If set, then remote activities that do not report progress for the specified period may be considered stalled and so retried.	Period
resizeClusterBeforeRunning	<p>Resize the cluster before performing this activity to accommodate DynamoDB data nodes specified as inputs or outputs.</p> <div data-bbox="502 931 633 967" style="border: 1px solid #ccc; padding: 5px; border-radius: 10px;">Note</div> <p>If your activity uses a <code>DynamoDBDataNode</code> as either an input or output data node, and if you set the <code>resizeClusterBeforeRunning</code> to <code>TRUE</code>, AWS Data Pipeline starts using <code>m3.xlarge</code> instance types. This overwrites your instance type choices with <code>m3.xlarge</code>, which could increase your monthly costs.</p>	Boolean
resizeClusterMaxInstances	A limit on the maximum number of instances that can be requested by the resize algorithm	Integer
retryDelay	The timeout duration between two retry attempts.	Period

Optional Fields	Description	Slot Type
scheduleType	<p>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the <code>ActivatePipeline</code> operation for each subsequent run. Values are: cron, ondemand, and timeseries.</p>	Enumeration

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnable ObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime

Runtime Fields	Description	Slot Type
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of the dependency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnable ObjectId"}
emrStepLog	Amazon EMR step logs available only on EMR activity attempts.	String
errord	The errord if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime

Runtime Fields	Description	Slot Type
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	The most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object.	DateTime
@scheduledStartTime	Schedule start time for object.	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref" : "myRunnableObject Id"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object.	String
@pipelineId	Id of the pipeline to which this object belongs to.	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Object.	String

See Also

- [ShellCommandActivity](#)
- [EmrActivity](#)

PigActivity

PigActivity provides native support for Pig scripts in AWS Data Pipeline without the requirement to use `ShellCommandActivity` or `EmrActivity`. In addition, PigActivity supports data staging. When the stage field is set to true, AWS Data Pipeline stages the input data as a schema in Pig without additional code from the user.

Example

The following example pipeline shows how to use PigActivity. The example pipeline performs the following steps:

- MyPigActivity1 loads data from Amazon S3 and runs a Pig script that selects a few columns of data and uploads it to Amazon S3.
- MyPigActivity2 loads the first output, selects a few columns and three rows of data, and uploads it to Amazon S3 as a second output.
- MyPigActivity3 loads the second output data, inserts two rows of data and only the column named "fifth" to Amazon RDS.
- MyPigActivity4 loads Amazon RDS data, selects the first row of data, and uploads it to Amazon S3.

```
{  
  "objects": [  
    {  
      "id": "MyInputData1",  
      "schedule": {  
        "ref": "MyEmrResourcePeriod"  
      },  
      "directoryPath": "s3://amzn-s3-demo-bucket/pigTestInput",  
      "name": "MyInputData1",  
      "dataFormat": {  
        "ref": "MyInputDataType1"  
      },  
      "script": "MyPigActivity1.pig"  
    },  
    {  
      "id": "MyOutputData1",  
      "schedule": {  
        "ref": "MyEmrResourcePeriod"  
      },  
      "directoryPath": "s3://amzn-s3-demo-bucket/pigTestOutput",  
      "name": "MyOutputData1",  
      "dataFormat": {  
        "ref": "MyOutputDataType1"  
      },  
      "script": "MyPigActivity2.pig"  
    },  
    {  
      "id": "MyOutputData2",  
      "schedule": {  
        "ref": "MyEmrResourcePeriod"  
      },  
      "directoryPath": "s3://amzn-s3-demo-bucket/pigTestOutput",  
      "name": "MyOutputData2",  
      "dataFormat": {  
        "ref": "MyOutputDataType2"  
      },  
      "script": "MyPigActivity3.pig"  
    },  
    {  
      "id": "MyOutputData3",  
      "schedule": {  
        "ref": "MyEmrResourcePeriod"  
      },  
      "directoryPath": "s3://amzn-s3-demo-bucket/pigTestOutput",  
      "name": "MyOutputData3",  
      "dataFormat": {  
        "ref": "MyOutputDataType3"  
      },  
      "script": "MyPigActivity4.pig"  
    }  
  ]  
}
```

```
"type": "S3DataNode"
},
{
  "id": "MyPigActivity4",
  "scheduleType": "CRON",
  "schedule": {
    "ref": "MyEmrResourcePeriod"
  },
  "input": {
    "ref": "MyOutputData3"
  },
  "pipelineLogUri": "s3://amzn-s3-demo-bucket/path/",
  "name": "MyPigActivity4",
  "runsOn": {
    "ref": "MyEmrResource"
  },
  "type": "PigActivity",
  "dependsOn": {
    "ref": "MyPigActivity3"
  },
  "output": {
    "ref": "MyOutputData4"
  },
  "script": "B = LIMIT ${input1} 1; ${output1} = FOREACH B GENERATE one;",
  "stage": "true"
},
{
  "id": "MyPigActivity3",
  "scheduleType": "CRON",
  "schedule": {
    "ref": "MyEmrResourcePeriod"
  },
  "input": {
    "ref": "MyOutputData2"
  },
  "pipelineLogUri": "s3://amzn-s3-demo-bucket/path",
  "name": "MyPigActivity3",
  "runsOn": {
    "ref": "MyEmrResource"
  },
  "type": "PigActivity",
  "dependsOn": {
    "ref": "MyPigActivity2"
  }
}
```

```
},
"output": {
    "ref": "MyOutputData3"
},
"stage": "true"
},
{
"id": "MyOutputData2",
"schedule": {
    "ref": "MyEmrResourcePeriod"
},
"name": "MyOutputData2",
"directoryPath": "s3://amzn-s3-demo-bucket/PigActivityOutput2",
"dataFormat": {
    "ref": "MyOutputDataType2"
},
"type": "S3DataNode"
},
{
"id": "MyOutputData1",
"schedule": {
    "ref": "MyEmrResourcePeriod"
},
"name": "MyOutputData1",
"directoryPath": "s3://amzn-s3-demo-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://amzn-s3-demo-bucket/PigActivityOutput3",
  "name": "MyOutputData4",
  "dataFormat": {
    "ref": "MyOutputDataType4"
  },
  "type": "S3DataNode"
},
{
  "id": "MyOutputDataType1",
  "name": "MyOutputDataType1",
  "column": [
    "First STRING",
    "Second STRING",
    "Third STRING",
    "Fourth STRING",
    "Fifth STRING"
  ]
}
```

```
"Fifth STRING",
"Sixth STRING",
"Seventh STRING",
"Eighth STRING"
],
"columnSeparator": "*",
"type": "Custom"
},
{
"id": "MyOutputData3",
"username": "___",
"schedule": {
"ref": "MyEmrResourcePeriod"
},
"insertQuery": "insert into #{table} (one) values (?)",
"name": "MyOutputData3",
"*password": "___",
"runsOn": {
"ref": "MyEmrResource"
},
"connectionString": "jdbc:mysql://example-database-instance:3306/example-
database",
"selectQuery": "select * from #{table}",
"table": "example-table-name",
"type": " MySqlDataNode"
},
{
"id": "MyOutputDataType2",
"name": "MyOutputDataType2",
"column": [
"Third STRING",
"Fourth STRING",
"Fifth STRING",
"Sixth STRING",
"Seventh STRING",
"Eighth STRING"
],
"type": "TSV"
},
{
"id": "MyPigActivity2",
"scheduleType": "CRON",
"schedule": {
"ref": "MyEmrResourcePeriod"
}
```

```
},
"input": {
    "ref": "MyOutputData1"
},
"pipelineLogUri": "s3://amzn-s3-demo-bucket/path",
"name": "MyPigActivity2",
"runsOn": {
    "ref": "MyEmrResource"
},
"dependsOn": {
    "ref": "MyPigActivity1"
},
"type": "PigActivity",
"script": "B = LIMIT ${input1} 3; ${output1} = FOREACH B GENERATE Third, Fourth, Fifth, Sixth, Seventh, Eighth;",
"output": {
    "ref": "MyOutputData2"
},
"stage": "true"
},
{
"id": "MyEmrResourcePeriod",
"startDateTime": "2013-05-20T00:00:00",
"name": "MyEmrResourcePeriod",
"period": "1 day",
"type": "Schedule",
"endDateTime": "2013-05-21T00:00:00"
},
{
"id": "MyPigActivity1",
"scheduleType": "CRON",
"schedule": {
    "ref": "MyEmrResourcePeriod"
},
"input": {
    "ref": "MyInputData1"
},
"pipelineLogUri": "s3://amzn-s3-demo-bucket/path",
"scriptUri": "s3://amzn-s3-demo-bucket/script/pigTestScript.q",
"name": "MyPigActivity1",
"runsOn": {
    "ref": "MyEmrResource"
},
"scriptVariable": [
```

```

    "column1=First",
    "column2=Second",
    "three=3"
],
{
  "type": "PigActivity",
  "output": {
    "ref": "MyOutputData1"
  },
  "stage": "true"
}
]
}

```

The content of `pigTestScript.q` is as follows.

```
B = LIMIT ${input1} $three; ${output1} = FOREACH B GENERATE $column1, $column2, Third,
Fourth, Fifth, Sixth, Seventh, Eighth;
```

Syntax

Object Invocation Fields	Description	Slot Type
schedule	<p>This object is invoked within the execution of a schedule interval. Users must specify a schedule reference to another object to set the dependency execution order for this object. Users can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"}. In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), users can create a parent object that has a schedule reference. For more information about example optional schedule configurations, see https://</p>	Reference Object, for example, "schedule": {"ref": "mySchedule"}

Object Invocation Fields	Description	Slot Type
	docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html	

Required Group (One of the following is required)	Description	Slot Type
script	The Pig script to run.	String
scriptUri	The location of the Pig script to run (for example, s3://scriptLocation).	String

Required Group (One of the following is required)	Description	Slot Type
runsOn	EMR Cluster on which this PigActivity runs.	Reference Object, for example, "runsOn": {"ref":"myEmrClusterId"}
workerGroup	The worker group. This is used for routing tasks. If you provide a runsOn value and workerGroup exists, workerGroup is ignored.	String

Optional Fields	Description	Slot Type
attemptStatus	The most recently reported status from the remote activity.	String
attemptTimeout	The timeout for remote work completion. If set, then a remote activity that does not complete within the set time of starting may be retried.	Period
dependsOn	Specifies the dependency on another runnable object.	Reference Object, for example, "dependsOn": {"ref": "myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun.	Enumeration
input	The input data source.	Reference Object, for example, "input": {"ref": "myDataNodeId"}
lateAfterTimeout	The elapsed time after pipeline start within which the object must complete. It is triggered only when the schedule type is not set to ondemand.	Period
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	The maximum number attempt retries on failure.	Integer
onFail	An action to run when current object fails.	Reference Object, for example, "onFail": {"ref": "myActionId"}

Optional Fields	Description	Slot Type
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, for example, "onLateAction": {"ref": "myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, for example, "onSuccess": {"ref": "myActionId"}
output	The output data source.	Reference Object, for example, "output": {"ref": "myDataNode Id"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, for example, "parent": {"ref": "myBaseObjId"}
pipelineLogUri	The Amazon S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
postActivityTaskConfig	Post-activity configuration script to be run. This consists of a URI of the shell script in Amazon S3 and a list of arguments.	Reference Object, for example, "postActivityTaskConfig": {"ref": "myShellScriptConfigId"}
preActivityTaskConfig	Pre-activity configuration script to be run. This consists of a URI of the shell script in Amazon S3 and a list of arguments.	Reference Object, for example, "preActivityTaskConfig": {"ref": "myShellScriptConfigId"}

Optional Fields	Description	Slot Type
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, for example, "precondition": {"ref": "myPreconditionId"}
reportProgressTimeout	The timeout for remote work successive calls to <code>reportProgress</code> . If set, then remote activities that do not report progress for the specified period may be considered stalled and so retried.	Period
resizeClusterBeforeRunning	Resize the cluster before performing this activity to accommodate DynamoDB data nodes specified as inputs or outputs.	Boolean
	<p>Note</p> <p>If your activity uses a <code>DynamoDBDataNode</code> as either an input or output data node, and if you set the <code>resizeClusterBeforeRunning</code> to TRUE, AWS Data Pipeline starts using <code>m3.xlarge</code> instance types. This overwrites your instance type choices with <code>m3.xlarge</code>, which could increase your monthly costs.</p>	
resizeClusterMaxInstances	A limit on the maximum number of instances that can be requested by the resize algorithm.	Integer
retryDelay	The timeout duration between two retry attempts.	Period

Optional Fields	Description	Slot Type
scheduleType	<p>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Time Series Style Scheduling means instances are scheduled at the end of each interval and Cron Style Scheduling means instances are scheduled at the beginning of each interval. An on-demand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use on-demand pipelines, you simply call the <code>ActivatePipeline</code> operation for each subsequent run. Values are: cron, ondemand, and timeseries.</p>	Enumeration
scriptVariable	The arguments to pass to the Pig script. You can use <code>scriptVariable</code> with <code>script</code> or <code>scriptUri</code> .	String
stage	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> .	Boolean

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, for example, "activeInstances": {"ref": "myRunnableObjectId"}

Runtime Fields	Description	Slot Type
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of the dependency chain the object failed on.	Reference Object, for example, "cascadeFailedOn": {"ref": "myRunnableObjectId"}
emrStepLog	Amazon EMR step logs available only on EMR activity attempts.	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime

Runtime Fields	Description	Slot Type
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for the object.	DateTime
@scheduledStartTime	Schedule start time for the object.	DateTime
@status	The status of this object.	String
@version	Pipeline version that the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, for example, "waitingOn":{"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object.	String

System Fields	Description	Slot Type
@pipelineId	ID of the pipeline to which this object belongs.	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects.	String

See Also

- [ShellCommandActivity](#)
- [EmrActivity](#)

RedshiftCopyActivity

Copies data from DynamoDB or Amazon S3 to Amazon Redshift. You can load data into a new table, or easily merge data into an existing table.

Here is an overview of a use case in which to use RedshiftCopyActivity:

1. Start by using AWS Data Pipeline to stage your data in Amazon S3.
2. Use RedshiftCopyActivity to move the data from Amazon RDS and Amazon EMR to Amazon Redshift.

This lets you load your data into Amazon Redshift where you can analyze it.

3. Use [SqlActivity](#) to perform SQL queries on the data that you've loaded into Amazon Redshift.

In addition, RedshiftCopyActivity let's you work with an S3DataNode, since it supports a manifest file. For more information, see [S3DataNode](#).

Example

The following is an example of this object type.

To ensure formats conversion, this example uses [EMPTYASNULL](#) and [IGNOREBLANKLINES](#) special conversion parameters in commandOptions. For information, see [Data Conversion Parameters](#) in the *Amazon Redshift Database Developer Guide*.

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

The following example pipeline definition shows an activity that uses the APPEND insert mode:

```
{  
  "objects": [  
    {  
      "id": "CSVID1",  
      "name": "DefaultCSV1",  
      "type": "CSV"  
    },  
    {  
      "id": "RedshiftDatabaseId1",  
      "databaseName": "dbname",  
      "username": "user",  
      "name": "DefaultRedshiftDatabase1",  
      "*password": "password",  
      "type": "RedshiftDatabase",  
      "clusterId": "redshiftclusterId"  
    },  
    {  
      "id": "Default",  
      "scheduleType": "timeseries",  
      "failureAndRerunMode": "CASCADE",  
      "name": "Default",  
      "role": "DataPipelineDefaultRole",  
      "resourceRole": "DataPipelineDefaultResourceRole"  
    },  
    {  
      "id": "RedshiftDataNodeId1",  
      "type": "RedshiftDataNode",  
      "parameters": {  
        "activity": "CSVID1",  
        "database": "RedshiftDatabaseId1",  
        "role": "Default",  
        "insertMode": "APPEND",  
        "commandOptions": ["EMPTYASNULL", "IGNOREBLANKLINES"]  
      }  
    }  
  ]  
}
```

```
"schedule": {
    "ref": "ScheduleId1"
},
"tableName": "orders",
"name": "DefaultRedshiftDataNode1",
"createTableSql": "create table StructuredLogs (requestBeginTime CHAR(30)
PRIMARY KEY DISTKEY SORTKEY, requestEndTime CHAR(30), hostname CHAR(100), requestDate
varchar(20));",
"type": "RedshiftDataNode",
"database": {
    "ref": "RedshiftDatabaseId1"
},
{
    "id": "Ec2ResourceId1",
    "schedule": {
        "ref": "ScheduleId1"
    },
    "securityGroups": "MySecurityGroup",
    "name": "DefaultEc2Resource1",
    "role": "DataPipelineDefaultRole",
    "logUri": "s3://myLogs",
    "resourceRole": "DataPipelineDefaultResourceRole",
    "type": "Ec2Resource"
},
{
    "id": "ScheduleId1",
    "startDateTime": "yyyy-mm-ddT00:00:00",
    "name": "DefaultSchedule1",
    "type": "Schedule",
    "period": "period",
    "endDateTime": "yyyy-mm-ddT00:00:00"
},
{
    "id": "S3DataNodeId1",
    "schedule": {
        "ref": "ScheduleId1"
    },
    "filePath": "s3://datapipeline-us-east-1/samples/hive-ads-samples.csv",
    "name": "DefaultS3DataNode1",
    "dataFormat": {
        "ref": "CSVId1"
    },
    "type": "S3DataNode"
```

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

APPEND operation adds items to a table regardless of the primary or sort keys. For example, if you have the following table, you can append a record with the same ID and user value.

ID(PK)	USER
1	aaa
2	bbb

You can append a record with the same ID and user value:

ID(PK)	USER
1	aaa
2	bbb
1	aaa

Note

If an APPEND operation is interrupted and retried, the resulting rerun pipeline potentially appends from the beginning. This may cause further duplication, so you should be aware of this behavior, especially if you have any logic that counts the number of rows.

For a tutorial, see [Copy Data to Amazon Redshift Using AWS Data Pipeline](#).

Syntax

Required Fields	Description	Slot Type
insertMode	<p>Determines what AWS Data Pipeline does with pre-existing data in the target table that overlaps with rows in the data to be loaded.</p> <p>Valid values are: KEEP_EXISTING , OVERWRITE_EXISTING , TRUNCATE and APPEND.</p> <p>KEEP_EXISTING adds new rows to the table, while leaving any existing rows unmodified.</p> <p>KEEP_EXISTING and OVERWRITE_EXISTING use the primary key, sort, and distribution keys to identify which incoming rows to match with existing rows. See Updating and Inserting New Data in the Amazon Redshift <i>Database Developer Guide</i>.</p> <p>TRUNCATE deletes all the data in the destination table before writing the new data.</p> <p>APPEND adds all records to the end of the Redshift table. APPEND does not require a primary, distribution key, or sort key so items that may be potential duplicates may be appended.</p>	Enumeration

Object Invocation Fields	Description	Slot Type
schedule	<p>This object is invoked within the execution of a schedule interval.</p> <p>Specify a schedule reference to another object to set the dependency execution order for this object.</p> <p>In most cases, we recommend to put the schedule reference on the default pipeline object so that all objects inherit that schedule. For example, you can explicitly set a schedule on the object by specifying "schedule": {"ref": "DefaultSchedule"} .</p> <p>If the master schedule in your pipeline contains nested schedules, create a parent object that has a schedule reference.</p> <p>For more information about example optional schedule configurations, see Schedule.</p>	Reference Object, such as: "schedule": {"ref": "myScheduleId"}

Required Group (One of the following is required)	Description	Slot Type
runsOn	The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.	Reference Object, e.g. "runsOn": {"ref": "myResourceId"}
workerGroup	The worker group. This is used for routing tasks. If you provide a runsOn value and	String

Required Group (One of the following is required)	Description	Slot Type
	workerGroup exists, workerGroup is ignored.	

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set, then a remote activity that does not complete within the set time of starting may be retried.	Period
commandOptions	<p>Takes parameters to pass to the Amazon Redshift data node during the COPY operation. For information on parameters, see COPY in the Amazon Redshift <i>Database Developer Guide</i>.</p> <p>As it loads the table, COPY attempts to implicitly convert the strings to the data type of the target column. In addition to the default data conversions that happen automatically, if you receive errors or have other conversion needs, you can specify additional conversion parameters. For information, see Data Conversion Parameter S in the Amazon Redshift <i>Database Developer Guide</i>.</p>	String

Optional Fields	Description	Slot Type
	<p>If a data format is associated with the input or output data node, then the provided parameters are ignored.</p> <p>Because the copy operation first uses COPY to insert data into a staging table, and then uses an INSERT command to copy the data from the staging table into the destination table, some COPY parameters do not apply, such as the COPY command's ability to enable automatic compression of the table. If compression is required, add column encoding details to the CREATE TABLE statement.</p> <p>Also, in some cases when it needs to unload data from the Amazon Redshift cluster and create files in Amazon S3, the RedshiftCopyActivity relies on the UNLOAD operation from Amazon Redshift.</p> <p>To improve performance during copying and unloading, specify PARALLEL OFF parameter from the UNLOAD command. For information on parameters, see UNLOAD in the Amazon Redshift <i>Database Developer Guide</i>.</p>	
dependsOn	Specify dependency on another runnable object.	Reference Object: "dependsOn": {"ref": "myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration

Optional Fields	Description	Slot Type
input	The input data node. The data source can be Amazon S3, DynamoDB, or Amazon Redshift.	Reference Object: "input":{ "ref":"my DataNodeId"}
lateAfterTimeout	The elapsed time after pipeline start within which the object must complete. It is triggered only when the schedule type is not set to ondemand.	Period
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object: "onFail": {"ref":"m yActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object: "onLateAction": {"ref":"myAc tionId"}
onSuccess	An action to run when current object succeeds.	Reference Object: "onSuccess": {"ref":"myActio nId"}
output	The output data node. The output location can be Amazon S3 or Amazon Redshift.	Reference Object: "output": {"ref":"m yDataNodeId"}

Optional Fields	Description	Slot Type
parent	Parent of the current object from which slots will be inherited.	Reference Object: "parent": {"ref":"myBaseObjectId"}
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object: "precondition": {"ref":"myPreconditionId"}
queue	<p>Corresponds to the <code>query_group</code> setting in Amazon Redshift, which allows you to assign and prioritize concurrent activities based on their placement in queues.</p> <p>Amazon Redshift limits the number of simultaneous connections to 15. For more information, see Assigning Queries to Queues in the <i>Amazon RDS Database Developer Guide</i>.</p>	String
reportProgressTimeout	<p>Timeout for remote work successive calls to <code>reportProgress</code>.</p> <p>If set, then remote activities that do not report progress for the specified period may be considered stalled and so retried.</p>	Period
retryDelay	The timeout duration between two retry attempts.	Period

Optional Fields	Description	Slot Type
scheduleType	<p>Allows you to specify whether the schedule for objects in your pipeline. Values are: <code>cron</code>, <code>ondemand</code>, and <code>timeseries</code> .</p> <p>The <code>timeseries</code> scheduling means instances are scheduled at the end of each interval.</p> <p>The <code>Cron</code> scheduling means instances are scheduled at the beginning of each interval.</p> <p>An ondemand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again.</p> <p>To use ondemand pipelines, call the <code>ActivatePipeline</code> operation for each subsequent run.</p> <p>If you use an ondemand schedule, you must specify it in the default object, and it must be the only <code>scheduleType</code> specified for objects in the pipeline.</p>	Enumeration

Optional Fields	Description	Slot Type
transformSql	<p>The SQL SELECT expression used to transform the input data.</p> <p>Run the <code>transformSql</code> expression on the table named <code>staging</code>.</p> <p>When you copy data from DynamoDB or Amazon S3, AWS Data Pipeline creates a table called "staging" and initially loads data in there. Data from this table is used to update the target table.</p> <p>The output schema of <code>transformSql</code> must match the final target table's schema.</p> <p>If you specify the <code>transformSql</code> option, 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.</p>	String

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object: "activeInstances": {"ref": "myRunnableObject", "id": "1234567890"} The reference object is defined in the Runtime Fields section .
@actualEndTime	Time when the execution of this object finished.	DateTime

Runtime Fields	Description	Slot Type
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of the dependency chain the object failed on.	Reference Object: "cascadeFailedOn": {"ref": "myRunnable ObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errord	The errord if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime

Runtime Fields	Description	Slot Type
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object.	DateTime
@scheduledStartTime	Schedule start time for object.	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object: "waitingOn": {"ref": "myRunnableObject.Id"}
System Fields	Description	Slot Type
@error	Error describing the ill-formed object.	String

System Fields	Description	Slot Type
@pipelineId	Id of the pipeline to which this object belongs to.	String
@sphere	The sphere of an object. Denotes its place in the life cycle. For example, Component Objects give rise to Instance Objects which execute Attempt Objects.	String

ShellCommandActivity

Runs a command or script. You can use `ShellCommandActivity` to run time-series or cron-like scheduled tasks.

When the `stage` field is set to true and used with an `S3DataNode`, `ShellCommandActivity` supports the concept of staging data, which means that you can move data from Amazon S3 to a stage location, such as Amazon EC2 or your local environment, perform work on the data using scripts and the `ShellCommandActivity`, and move it back to Amazon S3.

In this case, when your shell command is connected to an input `S3DataNode`, your shell scripts operate directly on the data using `INPUT1_STAGING_DIR`, `INPUT2_STAGING_DIR`, and other fields, 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 is a non-zero value.

Example

The following is an example of this object type.

{

```

"id" : "CreateDirectory",
"type" : "ShellCommandActivity",
"command" : "mkdir new-directory"
}

```

Syntax

Object Invocation Fields	Description	Slot Type
schedule	<p>This object is invoked within the execution of a schedule interval.</p> <p>To set the dependency execution order for this object, specify a schedule reference to another object.</p> <p>To satisfy this requirement, explicitly set a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"} .</p> <p>In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. If the pipeline consists of a tree of schedules (schedules within the master schedule), create a parent object that has a schedule reference.</p> <p>To spread the load, AWS Data Pipeline creates physical objects slightly ahead of schedule, but runs them on schedule.</p> <p>For more information about example optional schedule configurations, see https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html</p>	Reference Object, e.g. "schedule": {"ref": "myScheduleId"}

Required Group (One of the following is required)	Description	Slot Type
command	<p>The command to run. Use \$ to reference positional parameters and <code>scriptArgument</code> to specify the parameters for the command. This value and any associated parameters must function in the environment from which you are running the Task Runner.</p>	String
scriptUri	<p>An Amazon S3 URI path for a file to download and run as a shell command. Specify only one <code>scriptUri</code> , or <code>command</code> field. <code>scriptUri</code> cannot use parameters, use <code>command</code> instead.</p>	String

Required Group (One of the following is required)	Description	Slot Type
runsOn	<p>The computational resource to run the activity or command, for example, an Amazon EC2 instance or an Amazon EMR cluster.</p>	Reference Object, e.g. "runsOn":{"ref":"myResourceId"}
workerGroup	<p>Used for routing tasks. If you provide a <code>runsOn</code> value and <code>workerGroup</code> exists, <code>workerGroup</code> is ignored.</p>	String

Optional Fields	Description	Slot Type
attemptStatus	The most recently reported status from the remote activity.	String
attemptTimeout	The timeout for the remote work completion. If set, then a remote activity that does not complete within the specified starting time may be retried.	Period
dependsOn	Specifies a dependency on another runnable object.	Reference Object, e.g. "dependsOn": {"ref": "myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun.	Enumeration
input	The location of the input data.	Reference Object, e.g. "input": {"ref": "myDataNodeId"}
lateAfterTimeout	The elapsed time after pipeline start within which the object must complete. It is triggered only when the schedule type is not set to ondemand.	Period
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	The maximum number attempt retries on failure.	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref": "myActionId"}

Optional Fields	Description	Slot Type
onLateAction	Actions that should be triggered if an object has not yet been scheduled or is not completed.	Reference Object, e.g. "onLateAction": {"ref": "myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref": "myActionId"}
output	The location of the output data.	Reference Object, e.g. "output": {"ref": "myDataNodeId"}
parent	The parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref": "myBaseObjectId"}
pipelineLogUri	The Amazon S3 URI, such as 's3://BucketName/Key/' for uploading logs for the pipeline.	String
precondition	Optionally defines a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, e.g. "precondition": {"ref": "myPreconditionId"}
reportProgressTimeout	The timeout for successive calls to <code>reportProgress</code> by remote activities. If set, then remote activities that do not report progress for the specified period may be considered stalled and are retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period

Optional Fields	Description	Slot Type
scheduleType	<p>Allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of the interval or at the end of the interval.</p> <p>The values are: <code>cron</code>, <code>ondemand</code>, and <code>timeseries</code>.</p> <p>If set to <code>timeseries</code>, instances are scheduled at the end of each interval.</p> <p>If set to <code>Cron</code>, instances are scheduled at the beginning of each interval.</p> <p>If set to <code>ondemand</code>, you can run a pipeline one time, per activation. This means you do not have to clone or recreate the pipeline to run it again. If you use an ondemand schedule, specify it in the default object as the only <code>scheduleType</code> for objects in the pipeline. To use ondemand pipelines, call the <code>ActivatePipeline</code> operation for each subsequent run.</p>	Enumeration

Optional Fields	Description	Slot Type
scriptArgument	<p>A JSON-formatted array of strings to pass to the command specified by the command. For example, if command is echo \$1 \$2, specify scriptArgument as "param1", "param2". For multiple arguments and parameters, pass the scriptArgument as follows: "scriptArgument": "arg1", "scriptArgument": "param1", "scriptArgument": "arg2", "scriptArgument": "param2" . The scriptArgument can only be used with command; Using it with scriptUri causes an error.</p>	String
stage	<p>Determines whether staging is enabled and allows your shell commands to have access to the staged-data variables, such as \${INPUT1_STAGING_DIR} and \${OUTPUT1_STAGING_DIR} .</p>	Boolean
stderr	<p>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.</p>	String
stdout	<p>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.</p>	String

Runtime Fields	Description	Slot Type
@activeInstances	The list of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnable ObjectId"}
@actualEndTime	The time when the execution of this object finished.	DateTime
@actualStartTime	The time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	The description of the dependency chain that caused the object failure.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnable ObjectId"}
emrStepLog	Amazon EMR step logs available only on Amazon EMR activity attempts.	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which the object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for Amazon EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String

Runtime Fields	Description	Slot Type
@healthStatusFromInstanceId	The Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	The time at which the health status was updated last time.	DateTime
hostname	The host name of the client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	The time of the latest run for which the execution completed.	DateTime
@latestRunTime	The time of the latest run for which the execution was scheduled.	DateTime
@nextRunTime	The time of the run to be scheduled next.	DateTime
reportProgressTime	The most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	The schedule end time for object.	DateTime
@scheduledStartTime	The schedule start time for object.	DateTime
@status	The status of the object.	String
@version	The AWS Data Pipeline version used to create the object.	String
@waitingOn	The description of the list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn":{"ref":"myRunnableObject Id"}

System Fields	Description	Slot Type
@error	The error describing the ill-formed object.	String
@pipelineId	The Id of the pipeline to which this object belongs.	String
@sphere	The place of an object in the lifecycle. Component Objects give rise to Instance Objects which execute Attempt Objects.	String

See Also

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

SqlActivity

Runs an SQL query (script) on a database.

Example

The following is an example of this object type.

```
{  
  "id" : "MySqlActivity",  
  "type" : "SqlActivity",  
  "database" : { "ref": "MyDatabaseID" },  
  "script" : "SQLQuery" | "scriptUri" : s3://scriptBucket/query.sql,  
  "schedule" : { "ref": "MyScheduleID" },  
}  
}
```

Syntax

Required Fields	Description	Slot Type
database	The database on which to run the supplied SQL script.	Reference Object, e.g. "database": {"ref": "myDatabaseId"}

Object Invocation Fields	Description	Slot Type
schedule	<p>This object is invoked within the execution of a schedule interval. You must specify a schedule reference to another object to set the dependency execution order for this object. You can set a schedule explicitly on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"} .</p> <p>In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule.</p> <p>If the pipeline has a tree of schedules nested within the master schedule, create a parent object that has a schedule reference. For more information about example optional schedule configurations, see https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html</p>	Reference Object, e.g. "schedule": {"ref": "myScheduleId"}

Required Group (One of the following is required)	Description	Slot Type
script	The SQL script to run. You must specify <code>script</code> or <code>scriptUri</code> . When the script is stored in Amazon S3, then <code>script</code> is not evaluated as an expression. Specifying multiple values for <code>scriptArgument</code> is helpful when the script is stored in Amazon S3.	String
scriptUri	A URI specifying the location of an SQL script to execute in this activity.	String

Required Group (One of the following is required)	Description	Slot Type
runsOn	The computational resource to run the activity or command. For example, an Amazon EC2 instance or Amazon EMR cluster.	Reference Object, e.g. "runsOn": {"ref": "myResourceId"}
workerGroup	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.	String

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String

Optional Fields	Description	Slot Type
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
dependsOn	Specify dependency on another runnable object.	Reference Object, e.g. "dependsOn": {"ref": "myActivityId"}
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
input	Location of the input data.	Reference Object, e.g. "input": {"ref": "myDataNode1Id"}
lateAfterTimeout	The time period since the scheduled start of the pipeline within which the object run must start.	Period
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref": "myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed in the time period since the scheduled start of the pipeline as specified by 'lateAfterTimeout'.	Reference Object, e.g. "onLateAction": {"ref": "myActionId"}

Optional Fields	Description	Slot Type
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref": "myActionId"}
output	Location of the output data. This is only useful for referencing from within a script (for example <code>#{output.tablename}</code>) and for creating the output table by setting 'createTableSql' in the output data node. The output of the SQL query is not written to the output data node.	Reference Object, e.g. "output": {"ref": "myDataNodeId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref": "myBaseObjectId"}
pipelineLogUri	The S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
precondition	Optionally define a precondition. A data node is not marked "READY" until all preconditions have been met.	Reference Object, e.g. "precondition": {"ref": "myPreconditionId"}
queue	[Amazon Redshift only] Corresponds to the <code>query_group</code> setting in Amazon Redshift, which allows you to assign and prioritize concurrent activities based on their placement in queues. Amazon Redshift limits the number of simultaneous connections to 15. For more information, see Assigning Queries to Queues in the Amazon Redshift Database Developer Guide.	String

Optional Fields	Description	Slot Type
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set, then remote activities that do not report progress for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period
scheduleType	<p>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of interval or end of the interval. Values are: cron, ondemand, and timeseries .</p> <p>timeseries scheduling means instances are scheduled at the end of each interval.</p> <p>cron scheduling means instances are scheduled at the beginning of each interval.</p> <p>An ondemand schedule allows you to run a pipeline one time per activation. This means you do not have to clone or re-create the pipeline to run it again. If you use an ondemand schedule, it must be specified in the default object and must be the only scheduleType specified for objects in the pipeline. To use ondemand pipelines, call the ActivatePipeline operation for each subsequent run.</p>	Enumeration

Optional Fields	Description	Slot Type
scriptArgument	<p>A list of variables for the script. You can alternatively put expressions directly into the script field. Multiple values for scriptArgument are helpful when the script is stored in Amazon S3. Example: #{{format(@schedule dStartTime, "YY-MM-DD HH:MM:SS")}\n#{format(plusPeriod(@scheduledStartTime, "1 day"), "YY-MM-DD HH:MM:SS")}}</p>	String

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnable ObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of the dependency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnable ObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String

Runtime Fields	Description	Slot Type
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
@healthStatus	The health status of the object which reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime

Runtime Fields	Description	Slot Type
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref": "myRunnableObject Id"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object.	String
@pipelineId	Id of the pipeline to which this object belongs to.	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects.	String

Resources

The following are the AWS Data Pipeline resource objects:

Objects

- [Ec2Resource](#)
- [EmrCluster](#)
- [HttpProxy](#)

Ec2Resource

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

AWS Data Pipeline now supports IMDSv2 for the Amazon EC2 instance, which uses a session-oriented method to better handle authentication when retrieving metadata information from instances. A session begins and ends a series of requests that software running on an Amazon EC2 instance uses to access the locally-stored Amazon EC2 instance metadata and credentials. The software starts a session with a simple HTTP PUT request to IMDSv2. IMDSv2 returns a secret token to the software running on the Amazon EC2 instance, which will use the token as a password to make requests to IMDSv2 for metadata and credentials.

Note

To use IMDSv2 for your Amazon EC2 instance, you will need to modify the settings, as the default AMI is not compatible with IMDSv2. You can specify a new AMI version which you can retrieve via the following SSM parameter: `/aws/service/ami-amazon-linux-latest/amzn-ami-hvm-x86_64-ebs`.

For information about default Amazon EC2 instances that AWS Data Pipeline creates if you do not specify an instance, see [Default Amazon EC2 Instances by AWS Region](#).

Examples

EC2-Classic

Important

Only AWS accounts created before December 4, 2013 support the EC2-Classic platform. If you have one of these accounts, you may have the option to create Ec2Resource objects for a pipeline in an EC2-Classic network rather than a VPC. We strongly recommend that you create resources for all your pipelines in VPCs. In addition, if you have existing resources in EC2-Classic, we recommend that you migrate them to a VPC.

The following example object launches an EC2 instance into EC2-Classic, with some optional fields set.

```
{  
  "id" : "MyEC2Resource",  
  "type" : "Ec2Resource",  
  "actionOnTaskFailure" : "terminate",  
  "actionOnResourceFailure" : "retryAll",  
  "maximumRetries" : "1",  
  "instanceType" : "m5.large",  
  "securityGroups" : [  
    "test-group",  
    "default"  
,  
  "keyPair" : "my-key-pair"  
}
```

EC2-VPC

The following example object launches an EC2 instance into a nondefault VPC, with some optional fields set.

```
{  
  "id" : "MyEC2Resource",  
  "type" : "Ec2Resource",  
  "actionOnTaskFailure" : "terminate",  
  "actionOnResourceFailure" : "retryAll",  
  "maximumRetries" : "1",  
  "instanceType" : "m5.large",  
  "securityGroupIds" : [  
    "sg-12345678",  
    "sg-12345678"  
,  
  "subnetId": "subnet-12345678",  
  "associatePublicIpAddress": "true",  
  "keyPair" : "my-key-pair"  
}
```

Syntax

Required Fields	Description	Slot Type
resourceRole	The IAM role that controls the resources that the Amazon EC2 instance can access.	String

Required Fields	Description	Slot Type
role	The IAM role that AWS Data Pipeline uses to create the EC2 instance.	String

Object Invocation Fields	Description	Slot Type
schedule	<p>This object is invoked within the execution of a schedule interval.</p> <p>To set the dependency execution order for this object, specify a schedule reference to another object. You can do this in one of the following ways:</p> <ul style="list-style-type: none"> • To ensure that all objects in the pipeline inherit the schedule, set a schedule on the object explicitly: "schedule": {"ref": "DefaultSchedule"} . In most cases, it is useful to put the schedule reference on the default pipeline object, so that all objects inherit that schedule. • If the pipeline has schedules nested within the master schedule, you can create a parent object that has a schedule reference . For more information about example optional schedule configurations, see https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html 	Reference Object, for example "schedule": {"ref": "myScheduleId"}

Optional Fields	Description	Slot Type
actionOnResourceFailure	The action taken after a resource failure for this resource. Valid values are "retryall" and "retrynone" .	String
actionOnTaskFailure	The action taken after a task failure for this resource. Valid values are "continue" or "terminate" .	String
associatePublicIpAddress	Indicates whether to assign a public IP address to the instance. If the instance is in Amazon EC2 or Amazon VPC, the default value is true. Otherwise, the default value is false.	Boolean
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for the remote work completion. If set, then a remote activity that does not complete within the specified starting time may be retried.	Period
availabilityZone	The Availability Zone in which to launch the Amazon EC2 instance.	String
disableIMDSv1	Default value is false and enables both IMDSv1 and IMDSv2. If you set it to true then it disables IMDSv1 and only provides IMDSv2s	Boolean
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun.	Enumeration
httpProxy	The proxy host that clients use to connect to AWS services.	Reference Object, for example, "httpProxy": {"ref": "myHttpProxyId"}

Optional Fields	Description	Slot Type
imageId	<p>The ID of the AMI to use for the instance. By default, AWS Data Pipeline uses the HVM AMI virtualization type. The specific AMI IDs used are based on a Region. You can overwrite the default AMI by specifying the HVM AMI of your choice. For more information about AMI types, see Linux AMI Virtualization Types and Finding a Linux AMI in the <i>Amazon EC2 User Guide</i>.</p>	String
initTimeout	The amount of time to wait for the resource to start.	Period
instanceCount	Deprecated.	Integer
instanceType	The type of Amazon EC2 instance to start.	String
keyPair	The name of the key pair. If you launch an Amazon EC2 instance without specifying a key pair, you cannot log on to it.	String
lateAfterTimeout	The elapsed time after pipeline start within which the object must complete. It is triggered only when the schedule type is not set to ondemand.	Period
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	The maximum number of attempt retries on failure.	Integer
minInstanceCount	Deprecated.	Integer

Optional Fields	Description	Slot Type
onFail	An action to run when the current object fails.	Reference Object, for example "onFail": {"ref": "myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or is still running.	Reference Object, for example "onLateAction": {"ref": "myActionId"}
onSuccess	An action to run when the current object succeeds.	Reference Object, for example, "onSuccess": {"ref": "myActionId"}
parent	The parent of the current object from which slots are inherited.	Reference Object, for example, "parent": {"ref": "myBaseObjectID"}
pipelineLogUri	The Amazon S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
region	The code for the Region in which the Amazon EC2 instance should run. By default, the instance runs in the same Region as the pipeline. You can run the instance in the same Region as a dependent dataset.	Enumeration

Optional Fields	Description	Slot Type
reportProgressTimeout	The timeout for remote work successive calls to <code>reportProgress</code> . If set, then remote activities that do not report progress for the specified period may be considered stalled and will be retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period
runAsUser	The user to run the TaskRunner.	String
runsOn	This field is not allowed on this object.	Reference Object, for example, "runsOn": {"ref": "myResourceId"}

Optional Fields	Description	Slot Type
scheduleType	<p>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of an interval, at the end of the interval, or on demand.</p> <p>Values are:</p> <ul style="list-style-type: none"> • <code>timeseries</code> . Instances are scheduled at the end of each interval. • <code>cron</code>. Instances are scheduled at the beginning of each interval. • <code>ondemand</code>. Allows you to run a pipeline one time per activation. You do not have to clone or re-create the pipeline to run it again. If you use an on-demand schedule, it must be specified in the default object and must be the only <code>scheduleType</code> specified for objects in the pipeline. To use on-demand pipelines, call the <code>ActivatePipeline</code> operation for each subsequent run. 	Enumeration
securityGroupIds	The IDs of one or more Amazon EC2 security groups to use for the instances in the resource pool.	String
securityGroups	One or more Amazon EC2 security groups to use for the instances in the resource pool.	String
spotBidPrice	The maximum amount per hour for your Spot Instance in dollars, which is a decimal value between 0 and 20.00, exclusive.	String
subnetId	The ID of the Amazon EC2 subnet in which to start the instance.	String

Optional Fields	Description	Slot Type
terminateAfter	The number of hours after which to terminate the resource.	Period
useOnDemandOnLastAttempt	On the last attempt to request a Spot Instance, make a request for On-Demand Instances rather than a Spot Instance. This ensures that if all previous attempts have failed, the last attempt is not interrupted.	Boolean
workerGroup	This field is not allowed on this object.	String

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, for example, "activeInstances": {"ref": "myRunnable ObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of the dependency chain on which the object failed.	Reference Object, for example, "cascadeFailedOn": {"ref": "myRunnable ObjectId"}

Runtime Fields	Description	Slot Type
		yRunnable Object Id"}]
emrStepLog	Step logs are available only on Amazon EMR activity attempts.	String
errorId	The error ID if this object failed.	String
errorMessage	The error message if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@failureReason	The reason for the resource failure.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for Amazon EMR activities.	String
@healthStatus	The health status of the object that reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	Id of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime

Runtime Fields	Description	Slot Type
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	The most recent time that the remote activity reported progress.	DateTime
@scheduledEndTime	The schedule end time for the object.	DateTime
@scheduledStartTime	The schedule start time for the object.	DateTime
@status	The status of this object.	String
@version	The pipeline version with which the object was created.	String
@waitingOn	Description of the list of dependencies on which this object is waiting.	Reference Object, for example, "waitingOn": {"ref": "myRunnableObject", "object": "myRunnableObject"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object.	String
@pipelineId	ID of the pipeline to which this object belongs.	String
@sphere	The place of an object in the lifecycle. Component objects give rise to instance objects, which execute attempt objects.	String

EmrCluster

Represents the configuration of an Amazon EMR cluster. This object is used by [EmrActivity](#) and [HadoopActivity](#) to launch a cluster.

Contents

- [Schedulers](#)
- [Amazon EMR Release Versions](#)
- [Amazon EMR permissions](#)
- [Syntax](#)
- [Examples](#)
- [See Also](#)

Schedulers

Schedulers provide a way to specify resource allocation and job prioritization within a Hadoop cluster. Administrators or users can choose a scheduler for various classes of users and applications. A scheduler could use queues to allocate resources to users and applications. You set up those queues when you create the cluster. You can then set up priority for certain types of work and user over others. This provides for efficient use of cluster resources, while allowing more than one user to submit work to the cluster. There are three types of scheduler available:

- [FairScheduler](#) — Attempts to schedule resources evenly over a significant period of time.
- [CapacityScheduler](#) — Uses queues to allow cluster administrators to assign users to queues of varying priority and resource allocation.
- Default — Used by the cluster, which could be configured by your site.

Amazon EMR Release Versions

An Amazon EMR release is a set of open-source applications from the big data ecosystem. Each release comprises different big data applications, components, and features that you select to have Amazon EMR install and configure when you create a cluster. You specify the release version using the release label. Release labels are in the form `emr-x.x.x`. For example, `emr-5.30.0`. Amazon EMR clusters based on release label `emr-4.0.0` and later use the `releaseLabel` property to specify the release label of an `EmrCluster` object. Earlier versions use the `amiVersion` property.

Important

All Amazon EMR clusters created using release version 5.22.0 or later use [Signature Version 4](#) to authenticate requests to Amazon S3. Some earlier release versions use Signature Version 2. Signature Version 2 support is being discontinued. For more information, see [Amazon S3 Update – SigV2 Deprecation Period Extended and Modified](#). We strongly recommend that you use an Amazon EMR release version that supports Signature Version 4. For earlier version releases, beginning with EMR 4.7.x, the most recent release in the series has been updated to support Signature Version 4. When using an earlier version EMR release, we recommend that you use the latest release in the series. In addition, avoid releases earlier than EMR 4.7.0.

Considerations and Limitations

Use the latest version of Task Runner

If you are using a self-managed `EmrCluster` object with a release label, use the latest Task Runner. For more information about Task Runner, see [Working with Task Runner](#). You can configure property values for all Amazon EMR configuration classifications. For more information, see [Configuring Applications](#) in the *Amazon EMR Release Guide*, the [the section called "EmrConfiguration"](#), and [the section called "Property"](#) object references.

Support for IMDSv2

Earlier, AWS Data Pipeline supported only IMDSv1. Now, AWS Data Pipeline supports IMDSv2 in Amazon EMR 5.23.1, 5.27.1, and 5.32 or later, and Amazon EMR 6.2 or later. IMDSv2 uses a session-oriented method to better handle authentication when retrieving metadata information from instances. You should configure your instances to make IMDSv2 calls by creating user-managed resources using TaskRunner-2.0.

Amazon EMR 5.32 or later and Amazon EMR 6.x

The Amazon EMR 5.32 or later and 6.x release series uses Hadoop version 3.x, which introduced breaking changes in how Hadoop's classpath is evaluated as compared to Hadoop version 2.x. Common libraries like Joda-Time were removed from the classpath.

If [EmrActivity](#) or [HadoopActivity](#) runs a Jar file that has dependencies on a library that was removed in Hadoop 3.x, the step fails with the error `java.lang.NoClassDefFoundError` or

`java.lang.ClassNotFoundException`. This can happen for Jar files that ran with no issues using Amazon EMR 5.x release versions.

To fix the issue, you must copy Jar file dependencies to the Hadoop classpath on an `EmrCluster` object before starting the `EmrActivity` or the `HadoopActivity`. We provide a bash script to do this. The bash script is available in the following location, where `MyRegion` is the AWS Region where your `EmrCluster` object runs, for example `us-west-2`.

```
s3://datapipeline-MyRegion/MyRegion/bootstrap-actions/latest/TaskRunner/copy-jars-to-hadoop-classpath.sh
```

The way to run the script depends on whether `EmrActivity` or `HadoopActivity` runs on a resource managed by AWS Data Pipeline or runs on a self-managed resource.

If you use a resource managed by AWS Data Pipeline, add a `bootstrapAction` to the `EmrCluster` object. The `bootstrapAction` specifies the script and the Jar files to copy as arguments. You can add up to 255 `bootstrapAction` fields per `EmrCluster` object, and you can add a `bootstrapAction` field to an `EmrCluster` object that already has bootstrap actions.

To specify this script as a bootstrap action, use the following syntax, where `JarFileRegion` is the Region where the Jar file is saved, and each `MyJarFilen` is the absolute path in Amazon S3 of a Jar file to be copied to the Hadoop classpath. Do not specify Jar files that are in the Hadoop classpath by default.

```
s3://datapipeline-MyRegion/MyRegion/bootstrap-actions/latest/TaskRunner/copy-jars-to-hadoop-classpath.sh, JarFileRegion, MyJarFile1, MyJarFile2[, ...]
```

The following example specifies a bootstrap action that copies two Jar files in Amazon S3: `my-jar-file.jar` and the `emr-dynamodb-tool-4.14.0-jar-with-dependencies.jar`. The Region used in the example is `us-west-2`.

```
{  
  "id" : "MyEmrCluster",  
  "type" : "EmrCluster",  
  "keyPair" : "my-key-pair",  
  "masterInstanceType" : "m5.xlarge",  
  "coreInstanceType" : "m5.xlarge",  
  "coreInstanceCount" : "2",  
  "taskInstanceType" : "m5.xlarge",
```

```
"taskInstanceCount": "2",
"bootstrapAction" : ["s3://datapipeline-us-west-2/us-west-2/bootstrap-actions/
latest/TaskRunner/copy-jars-to-hadoop-classpath.sh,us-west-2,s3://path/to/my-jar-
file.jar,s3://dynamodb-dpl-us-west-2/emr-ddb-storage-handler/4.14.0/emr-dynamodb-
tools-4.14.0-jar-with-dependencies.jar"]
}
```

You must save and activate the pipeline for the change to the new bootstrapAction to take effect.

If you use a self-managed resource, you can download the script to the cluster instance and run it from the command line using SSH. The script creates a directory named /etc/hadoop/conf/shellprofile.d and a file named datapipeline-jars.sh in that directory. The jar files provided as command-line arguments are copied to a directory that the script creates named /home/hadoop/datapipeline_jars. If your cluster is set up differently, modify the script appropriately after downloading it.

The syntax for running the script on the command line is slightly different from using the bootstrapAction shown in the previous example. Use spaces instead of commas between arguments, as shown in the following example.

```
./copy-jars-to-hadoop-classpath.sh us-west-2 s3://path/to/my-jar-file.jar s3://
dynamodb-dpl-us-west-2/emr-ddb-storage-handler/4.14.0/emr-dynamodb-tools-4.14.0-jar-
with-dependencies.jar
```

Amazon EMR permissions

When you create a custom IAM role, carefully consider the minimum permissions necessary for your cluster to perform its work. Be sure to grant access to required resources, such as files in Amazon S3 or data in Amazon RDS, Amazon Redshift, or DynamoDB. If you wish to set `visibleToAllUsers` to `False`, your role must have the proper permissions to do so. Note that `DataPipelineDefaultRole` does not have these permissions. You must either provide a union of the `DefaultDataPipelineResourceRole` and `DataPipelineDefaultRole` roles as the `EmrCluster` object role, or create your own role for this purpose.

Syntax

Object Invocation Fields	Description	Slot Type
schedule	<p>This object is invoked within the execution of a schedule interval. Specify a schedule reference to another object to set the dependency execution order for this object. You can satisfy this requirement by explicitly setting a schedule on the object, for example, by specifying "schedule": {"ref": "DefaultSchedule"} . In most cases, it is better to put the schedule reference on the default pipeline object so that all objects inherit that schedule. Or, if the pipeline has a tree of schedules (schedules within the master schedule), you can create a parent object that has a schedule reference. For more information about example optional schedule configurations, see https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html</p>	Reference Object, for example, "schedule": {"ref": "myScheduleId"}

Optional Fields	Description	Slot Type
actionOnResourceFailure	The action taken after a resource failure for this resource. Valid values are "retryall", which retries all tasks to the cluster for the specified duration, and "retrynone".	String
actionOnTaskFailure	The action taken after task failure for this resource. Valid values are "continue", meaning do not terminate the cluster, and "terminate."	String

Optional Fields	Description	Slot Type
additionalMasterSecurityGroupIds	The identifier of additional master security groups of the EMR cluster, which follows the form sg-01XXXX6a. For more information, see Amazon EMR Additional Security Groups in the <i>Amazon EMR Management Guide</i> .	String
additionalSlaveSecurityGroupIds	The identifier of additional slave security groups of the EMR cluster, which follows the form sg-01XXXX6a .	String
amiVersion	The Amazon Machine Image (AMI) version that Amazon EMR uses to install the cluster nodes. For more information, see the Amazon EMR Management Guide .	String
applications	Applications to install in the cluster with comma-separated arguments. By default, Hive and Pig are installed. This parameter is applicable only for Amazon EMR version 4.0 and later.	String
attemptStatus	The most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set, then a remote activity that does not complete within the set time of starting may be retried.	Period
availabilityZone	The Availability Zone in which to run the cluster.	String

Optional Fields	Description	Slot Type
bootstrapAction	<p>An action to run when the cluster starts. You can specify comma-separated arguments. To specify multiple actions, up to 255, add multiple <code>bootstrapAction</code> fields. The default behavior is to start the cluster without any bootstrap actions.</p>	String
configuration	<p>Configuration for the Amazon EMR cluster. This parameter is applicable only for Amazon EMR version 4.0 and later.</p>	Reference Object, for example, "configuration": {"ref": "myEmrConfigurationId"}
coreInstanceBidPrice	<p>The maximum Spot price your are willing to pay for Amazon EC2 instances. If a bid price is specified, Amazon EMR uses Spot Instances for the instance group. Specified in USD.</p>	String
coreInstanceCount	<p>The number of core nodes to use for the cluster.</p>	Integer
coreInstanceType	<p>The type of Amazon EC2 instance to use for core nodes. See Supported Amazon EC2 Instances for Amazon EMR Clusters.</p>	String
coreGroupConfiguration	<p>The configuration for the Amazon EMR cluster core instance group. This parameter is applicable only for Amazon EMR version 4.0 and later.</p>	Reference Object, for example "configuration": {"ref": "myEmrConfigurationId"}

Optional Fields	Description	Slot Type
coreEbsConfiguration	<p>The configuration for Amazon EBS volumes that will be attached to each of the core nodes in the core group in the Amazon EMR cluster.</p> <p>For more information, see Instance Types That Support EBS Optimization in the <i>Amazon EC2 User Guide</i>.</p>	Reference Object, for example “coreEbsConfiguration”: {"ref": "myEbsConfiguration"}
customAmiId	<p>Applies only to Amazon EMR release version 5.7.0 and later. Specifies the AMI ID of a custom AMI to use when Amazon EMR provisions Amazon EC2 instances. It can also be used instead of bootstrap actions to customize cluster node configurations. For more information, see the following topic in the Amazon EMR Management Guide. Using a custom AMI</p>	String
EbsBlockDeviceConfig	<p>The configuration of a requested Amazon EBS block device associated with the instance group. Includes a specified number of volumes that will be associated with each instance in the instance group. Includes <code>volumesPerInstance</code> and <code>volumeSpecification</code>, where:</p> <ul style="list-style-type: none"> • <code>volumesPerInstance</code> is the number of EBS volumes with a specific volume configuration that will be associated with each instance in the instance group. • <code>volumeSpecification</code> is the Amazon EBS volume specifications, such as volume type, IOPS, and size in Gigabytes (GiB) that will be requested for the EBS volume attached to an EC2 instance in the Amazon EMR cluster. 	Reference Object, for example “EbsBlockDeviceConfig”: {"ref": "myEbsBlockDeviceConfig"}

Optional Fields	Description	Slot Type
emrManageDMasterSecurityGroupId	The identifier of the master security group of the Amazon EMR cluster, which follows the form of sg-01XXXX6a . For more information, see Configure Security Groups in the <i>Amazon EMR Management Guide</i> .	String
emrManagedSlaveSecurityGroupId	The identifier of the slave security group of the Amazon EMR cluster, which follows the form sg-01XXXX6a .	String
enableDebugging	Enables debugging on the Amazon EMR cluster.	String
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun.	Enumeration
hadoopSchedulerType	The scheduler type of the cluster. Valid types are: PARALLEL_FAIR_SCHEDULING , PARALLEL_CAPACITY_SCHEDULING , and DEFAULT_SCHEDULER .	Enumeration
httpProxy	The proxy host that clients use to connect to AWS services.	Reference Object, for example, "httpProxy":{"ref":"myHttpProxyId"}
initTimeout	The amount of time to wait for the resource to start.	Period
keyPair	The Amazon EC2 key pair to use to log on to the master node of the Amazon EMR cluster.	String
lateAfterTimeout	The elapsed time after pipeline start within which the object must complete. It is triggered only when the schedule type is not set to ondemand.	Period

Optional Fields	Description	Slot Type
masterInstanceBidPrice	The maximum Spot price you are willing to pay for Amazon EC2 instances. It is a decimal value between 0 and 20.00, exclusive. Specified in USD. Setting this value enables Spot Instances for the Amazon EMR cluster master node. If a bid price is specified, Amazon EMR uses Spot Instances for the instance group.	String
masterInstanceType	The type of Amazon EC2 instance to use for the master node. See Supported Amazon EC2 Instances for Amazon EMR Clusters .	String
masterGroupConfiguration	The configuration for the Amazon EMR cluster master instance group. This parameter is applicable only for Amazon EMR version 4.0 and later.	Reference Object, for example "configuration": {"ref": "myEmrConfigurationId"}
masterEbsConfiguration	The configuration for Amazon EBS volumes that will be attached to each of the master nodes in the master group in the Amazon EMR cluster. For more information, see Instance Types That Support EBS Optimization in the Amazon EC2 User Guide .	Reference Object, for example "masterEbsConfiguration": {"ref": "myEbsConfiguration"}
maxActiveInstances	The maximum number of concurrent active instances of a component. Re-runs do not count toward the number of active instances.	Integer
maximumRetries	Maximum number attempt retries on failure.	Integer
onFail	An action to run when the current object fails.	Reference Object, for example, "onFail": {"ref": "myActionId"}

Optional Fields	Description	Slot Type
onLateAction	Actions that should be triggered if an object has not yet been scheduled or is still not completed.	Reference Object, for example, "onLateAction": {"ref": "myActionId"}
onSuccess	An action to run when the current object succeeds.	Reference Object, for example, "onSuccess": {"ref": "myActionId"}
parent	Parent of the current object from which slots are inherited.	Reference Object, for example. "parent": {"ref": "myBaseObject"}
pipelineLogUri	The Amazon S3 URI (such as 's3://BucketName/Key/') for uploading logs for the pipeline.	String
region	The code for the region that the Amazon EMR cluster should run in. By default, the cluster runs in the same region as the pipeline. You can run the cluster in the same region as a dependent dataset.	Enumeration
releaseLabel	Release label for the EMR cluster.	String
reportProgressTimeout	Timeout for remote work successive calls to <code>reportProgress</code> . If set, then remote activities that do not report progress for the specified period may be considered stalled and so retried.	Period

Optional Fields	Description	Slot Type
resourceRole	The IAM role that AWS Data Pipeline uses to create the Amazon EMR cluster. The default role is <code>DataPipelineDefaultRole</code> .	String
retryDelay	The timeout duration between two retry attempts.	Period
role	The IAM role passed to Amazon EMR to create EC2 nodes.	String
runsOn	This field is not allowed on this object.	Reference Object, for example, "runsOn": {"ref": "myResourceId"}
securityConfiguration	The identifier of the EMR security configuration that will be applied to the cluster. This parameter is applicable only for Amazon EMR version 4.8.0 and later.	String
serviceAccessSecurityGroupId	The identifier for the service access security group of the Amazon EMR cluster.	String. It follows the form of sg-01XXXX6a , for example, sg-1234abcd .

Optional Fields	Description	Slot Type
scheduleType	<p>Schedule type allows you to specify whether the objects in your pipeline definition should be scheduled at the beginning of the interval, or end of the interval. Values are: <code>cron</code>, <code>ondemand</code>, and <code>timeseries</code> . The <code>timeseries</code> scheduling means that instances are scheduled at the end of each interval. The <code>cron</code> scheduling means that instances are scheduled at the beginning of each interval. An <code>ondemand</code> schedule allows you to run a pipeline one time per activation. You do not have to clone or re-create the pipeline to run it again. If you use an <code>ondemand</code> schedule, it must be specified in the <code>default</code> object and must be the only <code>scheduleType</code> specified for objects in the pipeline. To use <code>ondemand</code> pipelines, call the <code>ActivatePipeline</code> operation for each subsequent run.</p>	Enumeration
subnetId	The identifier of the subnet into which to launch the Amazon EMR cluster.	String
supportedProducts	A parameter that installs third-party software on an Amazon EMR cluster, for example, a third-party distribution of Hadoop.	String
taskInstanceBidPrice	The maximum Spot price your are willing to pay for EC2 instances. A decimal value between 0 and 20.00, exclusive. Specified in USD. If a bid price is specified, Amazon EMR uses Spot Instances for the instance group.	String
taskInstanceCount	The number of task nodes to use for the Amazon EMR cluster.	Integer

Optional Fields	Description	Slot Type
taskInstanceType	The type of Amazon EC2 instance to use for task nodes.	String
taskGroupConfiguration	The configuration for the Amazon EMR cluster task instance group. This parameter is applicable only for Amazon EMR version 4.0 and later.	Reference Object, for example "configuration": {"ref": "myEmrConfigurationId"}
taskEbsConfiguration	<p>The configuration for Amazon EBS volumes that will be attached to each of the task nodes in the task group in the Amazon EMR cluster.</p> <p>For more information, see Instance Types That Support EBS Optimization in the <i>Amazon EC2 User Guide</i>.</p>	Reference Object, for example "taskEbsConfiguration": {"ref": "myEbsConfiguration"}
terminateAfter	Terminate the resource after these many hours.	Integer

Optional Fields	Description	Slot Type
VolumeSpecification	<p>The Amazon EBS volume specifications, such as volume type, IOPS, and size in Gigabytes (GiB) that will be requested for the Amazon EBS volume attached to an Amazon EC2 instance in the Amazon EMR cluster. The node can be a core, master or task node.</p> <p>The VolumeSpecification includes:</p> <ul style="list-style-type: none"> • <code>iops()</code> Integer. The number of I/O operations per second (IOPS) that the Amazon EBS volume supports, for example, 1000. For more information, see EBS I/O Characteristics in the <i>Amazon EC2 User Guide</i>. • <code>sizeinGB()</code> . Integer. The Amazon EBS volume size, in gibibytes (GiB), for example 500. For information about valid combinations of volume types and hard drive sizes, see EBS Volume Types in the <i>Amazon EC2 User Guide</i>. • <code>volumeType</code> . String. The Amazon EBS volume type, for example, gp2. The supported volume types include standard, gp2, io1, st1, sc1, and others. For more information, see EBS Volume Types in the <i>Amazon EC2 User Guide</i>. 	Reference Object, for example "VolumeSpecification": {"ref": "myVolume Specification"}
useOnDemandOnLastAttempt	On the last attempt to request a resource, make a request for On-Demand Instances rather than Spot Instances. This ensures that if all previous attempts have failed, the last attempt is not interrupted.	Boolean
workerGroup	Field not allowed on this object.	String

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, for example, "activeInstances": {"ref": "myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of the dependency chain on which the object failed.	Reference Object, for example, "cascadeFailedOn": {"ref": "myRunnableObjectId"}
emrStepLog	Step logs available only on Amazon EMR activity attempts.	String
errorId	The error ID if this object failed.	String
errorMessage	The error message if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
@failureReason	The reason for the resource failure.	String
@finishedTime	The time at which this object finished its execution.	DateTime
hadoopJobLog	Hadoop job logs available on attempts for Amazon EMR activities.	String

Runtime Fields	Description	Slot Type
@healthStatus	The health status of the object that reflects success or failure of the last object instance that reached a terminated state.	String
@healthStatusFromInstanceId	ID of the last instance object that reached a terminated state.	String
@healthStatusUpdatedTime	Time at which the health status was updated last time.	DateTime
hostname	The host name of client that picked up the task attempt.	String
@lastDeactivatedTime	The time at which this object was last deactivated.	DateTime
@latestCompletedRunTime	Time the latest run for which the execution completed.	DateTime
@latestRunTime	Time the latest run for which the execution was scheduled.	DateTime
@nextRunTime	Time of run to be scheduled next.	DateTime
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object.	DateTime
@scheduledStartTime	Schedule start time for object.	DateTime
@status	The status of this object.	String
@version	Pipeline version with which the object was created.	String

Runtime Fields	Description	Slot Type
@waitingOn	Description of the list of dependencies on which this object is waiting.	Reference Object, for example, "waitingOn": {"ref": "myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object.	String
@pipelineId	ID of the pipeline to which this object belongs.	String
@sphere	The place of an object in the lifecycle. Component objects give rise to instance objects, which execute attempt objects.	String

Examples

The following are examples of this object type.

Contents

- [Launch an Amazon EMR cluster with hadoopVersion](#)
- [Launch an Amazon EMR cluster with release label emr-4.x or greater](#)
- [Install additional software on your Amazon EMR cluster](#)
- [Disable server-side encryption on 3.x releases](#)
- [Disable server-side encryption on 4.x releases](#)
- [Configure Hadoop KMS ACLs and create encryption zones in HDFS](#)
- [Specify custom IAM roles](#)
- [Use EmrCluster Resource in AWS SDK for Java](#)
- [Configure an Amazon EMR cluster in a private subnet](#)
- [Attach EBS volumes to cluster nodes](#)

Launch an Amazon EMR cluster with hadoopVersion

Example

The following example launches an Amazon EMR cluster using AMI version 1.0 and Hadoop 0.20.

```
{  
    "id" : "MyEmrCluster",  
    "type" : "EmrCluster",  
    "hadoopVersion" : "0.20",  
    "keyPair" : "my-key-pair",  
    "masterInstanceType" : "m3.xlarge",  
    "coreInstanceType" : "m3.xlarge",  
    "coreInstanceCount" : "10",  
    "taskInstanceType" : "m3.xlarge",  
    "taskInstanceCount": "10",  
    "bootstrapAction" : ["s3://Region.elasticmapreduce/bootstrap-actions/configure-  
    hadoop,arg1,arg2,arg3","s3://Region.elasticmapreduce/bootstrap-actions/configure-  
    hadoop/configure-other-stuff,arg1,arg2"]  
}
```

Launch an Amazon EMR cluster with release label emr-4.x or greater

Example

The following example launches an Amazon EMR cluster using the newer releaseLabel field:

```
{  
    "id" : "MyEmrCluster",  
    "type" : "EmrCluster",  
    "keyPair" : "my-key-pair",  
    "masterInstanceType" : "m3.xlarge",  
    "coreInstanceType" : "m3.xlarge",  
    "coreInstanceCount" : "10",  
    "taskInstanceType" : "m3.xlarge",  
    "taskInstanceCount": "10",  
    "releaseLabel": "emr-4.1.0",  
    "applications": ["spark", "hive", "pig"],  
    "configuration": {"ref":"myConfiguration"}  
}
```

Install additional software on your Amazon EMR cluster

Example

`EmrCluster` provides the `supportedProducts` field that installs third-party software on an Amazon EMR cluster, for example, it lets you install a custom distribution of Hadoop, such as MapR. It accepts a comma-separated list of arguments for the third-party software to read and act on. The following example shows how to use the `supportedProducts` field of `EmrCluster` to create a custom MapR M3 edition cluster with Karmasphere Analytics installed, and run an `EmrActivity` object on it.

```
{
    "id": "MyEmrActivity",
    "type": "EmrActivity",
    "schedule": {"ref": "ResourcePeriod"},
    "runsOn": {"ref": "MyEmrCluster"},
    "postStepCommand": "echo Ending job >> /mnt/var/log/stepCommand.txt",
    "preStepCommand": "echo Starting job > /mnt/var/log/stepCommand.txt",
    "step": "/home/hadoop/contrib/streaming/hadoop-streaming.jar,-input,s3n://
elasticmapreduce/samples/wordcount/input,-output, \
        hdfs://output32113/-mapper,s3n://elasticmapreduce/samples/wordcount/
wordSplitter.py,-reducer,aggregate"
},
{
    "id": "MyEmrCluster",
    "type": "EmrCluster",
    "schedule": {"ref": "ResourcePeriod"},
    "supportedProducts": ["mapr,--edition,m3,--version,1.2,--key1,value1","karmasphere-
enterprise-utility"],
    "masterInstanceType": "m3.xlarge",
    "taskInstanceType": "m3.xlarge"
}
```

Disable server-side encryption on 3.x releases

Example

An `EmrCluster` activity with a Hadoop version 2.x created by AWS Data Pipeline enables server-side encryption by default. If you would like to disable server-side encryption, you must specify a bootstrap action in the cluster object definition.

The following example creates an `EmrCluster` activity with server-side encryption disabled:

```
{  
    "id": "NoSSEEmrCluster",  
    "type": "EmrCluster",  
    "hadoopVersion": "2.x",  
    "keyPair": "my-key-pair",  
    "masterInstanceType": "m3.xlarge",  
    "coreInstanceType": "m3.large",  
    "coreInstanceCount": "10",  
    "taskInstanceType": "m3.large",  
    "taskInstanceCount": "10",  
    "bootstrapAction": ["s3://Region.elasticmapreduce/bootstrap-actions/configure-  
hadoop,-e, fs.s3.enableServerSideEncryption=false"]  
}
```

Disable server-side encryption on 4.x releases

Example

You must disable server-side encryption using a `EmrConfiguration` object.

The following example creates an `EmrCluster` activity with server-side encryption disabled:

```
{  
    "name": "ReleaseLabelCluster",  
    "releaseLabel": "emr-4.1.0",  
    "applications": ["spark", "hive", "pig"],  
    "id": "myResourceId",  
    "type": "EmrCluster",  
    "configuration": {  
        "ref": "disableSSE"  
    }  
,  
{  
    "name": "disableSSE",  
    "id": "disableSSE",  
    "type": "EmrConfiguration",  
    "classification": "emrfs-site",  
    "property": [{  
        "ref": "enableServerSideEncryption"  
    }  
]  
,  
{
```

```
"name": "enableServerSideEncryption",
"id": "enableServerSideEncryption",
"type": "Property",
"key": "fs.s3.enableServerSideEncryption",
"value": "false"
}
```

Configure Hadoop KMS ACLs and create encryption zones in HDFS

Example

The following objects create ACLs for Hadoop KMS and create encryption zones and corresponding encryption keys in HDFS:

```
{
  "name": "kmsAcls",
  "id": "kmsAcls",
  "type": "EmrConfiguration",
  "classification": "hadoop-kms-acls",
  "property": [
    {"ref": "kmsBlacklist"},
    {"ref": "kmsAcl"}
  ],
},
{
  "name": "hdfsEncryptionZone",
  "id": "hdfsEncryptionZone",
  "type": "EmrConfiguration",
  "classification": "hdfs-encryption-zones",
  "property": [
    {"ref": "hdfsPath1"},
    {"ref": "hdfsPath2"}
  ],
},
{
  "name": "kmsBlacklist",
  "id": "kmsBlacklist",
  "type": "Property",
  "key": "hadoop.kms.blacklist.CREATE",
  "value": "foo,myBannedUser"
},
{
  "name": "kmsAcl",
  "id": "kmsAcl",
}
```

```
"type": "Property",
"key": "hadoop.kms.acl.ROLLOVER",
"value": "myAllowedUser"
},
{
  "name": "hdfsPath1",
  "id": "hdfsPath1",
  "type": "Property",
  "key": "/myHDFSPath1",
  "value": "path1_key"
},
{
  "name": "hdfsPath2",
  "id": "hdfsPath2",
  "type": "Property",
  "key": "/myHDFSPath2",
  "value": "path2_key"
}
```

Specify custom IAM roles

Example

By default, AWS Data Pipeline passes `DataPipelineDefaultRole` as the Amazon EMR service role and `DataPipelineDefaultResourceRole` as the Amazon EC2 instance profile to create resources on your behalf. However, you can create a custom Amazon EMR service role and a custom instance profile and use them instead. AWS Data Pipeline should have sufficient permissions to create clusters using the custom role, and you must add AWS Data Pipeline as a trusted entity.

The following example object specifies custom roles for the Amazon EMR cluster:

```
{
  "id": "MyEmrCluster",
  "type": "EmrCluster",
  "hadoopVersion": "2.x",
  "keyPair": "my-key-pair",
  "masterInstanceType": "m3.xlarge",
  "coreInstanceType": "m3.large",
  "coreInstanceCount": "10",
  "taskInstanceType": "m3.large",
  "taskInstanceCount": "10",
  "role": "emrServiceRole",
  "resourceRole": "emrInstanceProfile"
```

}

Use EmrCluster Resource in AWS SDK for Java

Example

The following example shows how to use an `EmrCluster` and `EmrActivity` to create an Amazon EMR 4.x cluster to run a Spark step using the Java SDK:

```
public class dataPipelineEmr4 {

    public static void main(String[] args) {

        AWS Credentials credentials = null;
        credentials = new ProfileCredentialsProvider("/path/to/
AwsCredentials.properties","default").getCredentials();
        DataPipelineClient dp = new DataPipelineClient(credentials);
        CreatePipelineRequest createPipeline = new
CreatePipelineRequest().withName("EMR4SDK").withUniqueId("unique");
        CreatePipelineResult createPipelineResult = dp.createPipeline(createPipeline);
        String pipelineId = createPipelineResult.getPipelineId();

        PipelineObject emrCluster = new PipelineObject()
            .withName("EmrClusterObj")
            .withId("EmrClusterObj")
            .withFields(
                new Field().withKey("releaseLabel").withStringValue("emr-4.1.0"),
                new Field().withKey("coreInstanceCount").withStringValue("3"),
                new Field().withKey("applications").withStringValue("spark"),
                new Field().withKey("applications").withStringValue("Presto-Sandbox"),
                new Field().withKey("type").withStringValue("EmrCluster"),
                new Field().withKey("keyPair").withStringValue("myKeyName"),
                new Field().withKey("masterInstanceType").withStringValue("m3.xlarge"),
                new Field().withKey("coreInstanceType").withStringValue("m3.xlarge")
            );

        PipelineObject emrActivity = new PipelineObject()
            .withName("EmrActivityObj")
            .withId("EmrActivityObj")
            .withFields(
                new Field().withKey("step").withStringValue("command-runner.jar,spark-submit,--
executor-memory,1g,--class,org.apache.spark.examples.SparkPi,/usr/lib/spark/lib/spark-
examples.jar,10"),
            );
    }
}
```

```
new Field().withKey("runsOn").withRefValue("EmrClusterObj"),
new Field().withKey("type").withStringValue("EmrActivity")
);

PipelineObject schedule = new PipelineObject()
    .withName("Every 15 Minutes")
    .withId("DefaultSchedule")
    .withFields(
        new Field().withKey("type").withStringValue("Schedule"),
        new Field().withKey("period").withStringValue("15 Minutes"),
        new Field().withKey("startAt").withStringValue("FIRST_ACTIVATION_DATE_TIME")
    );

PipelineObject defaultObject = new PipelineObject()
    .withName("Default")
    .withId("Default")
    .withFields(
        new Field().withKey("failureAndRerunMode").withStringValue("CASCADE"),
        new Field().withKey("schedule").withRefValue("DefaultSchedule"),
        new
Field().withKey("resourceRole").withStringValue("DataPipelineDefaultResourceRole"),
        new Field().withKey("role").withStringValue("DataPipelineDefaultRole"),
        new Field().withKey("pipelineLogUri").withStringValue("s3://myLogUri"),
        new Field().withKey("scheduleType").withStringValue("cron")
    );

List<PipelineObject> pipelineObjects = new ArrayList<PipelineObject>();

pipelineObjects.add(emrActivity);
pipelineObjects.add(emrCluster);
pipelineObjects.add(defaultObject);
pipelineObjects.add(schedule);

PutPipelineDefinitionRequest putPipelineDefintion = new PutPipelineDefinitionRequest()
    .withPipelineId(pipelineId)
    .withPipelineObjects(pipelineObjects);

PutPipelineDefinitionResult putPipelineResult =
dp.putPipelineDefinition(putPipelineDefintion);
System.out.println(putPipelineResult);

ActivatePipelineRequest activatePipelineReq = new ActivatePipelineRequest()
    .withPipelineId(pipelineId);
ActivatePipelineResult activatePipelineRes = dp.activatePipeline(activatePipelineReq);
```

```
        System.out.println(activatePipelineRes);
        System.out.println(pipelineId);

    }

}
```

Configure an Amazon EMR cluster in a private subnet

Example

This example includes a configuration that launches the cluster into a private subnet in a VPC. For more information, see [Launch Amazon EMR Clusters into a VPC](#) in the *Amazon EMR Management Guide*. This configuration is optional. You can use it in any pipeline that uses an `EmrCluster` object.

To launch an Amazon EMR cluster in a private subnet, specify `SubnetId`, `emrManagedMasterSecurityGroupId`, `emrManagedSlaveSecurityGroupId`, and `serviceAccessSecurityGroupId` in your `EmrCluster` configuration.

```
{
  "objects": [
    {
      "output": {
        "ref": "S3BackupLocation"
      },
      "input": {
        "ref": "DDBSourceTable"
      },
      "maximumRetries": "2",
      "name": "TableBackupActivity",
      "step": "s3://dynamodb-emr-#{myDDBRegion}/emr-ddb-storage-handler/2.1.0/emr-
ddb-2.1.0.jar,org.apache.hadoop.dynamodb.tools.DynamoDbExport,#{output.directoryPath},#{input.t
      "id": "TableBackupActivity",
      "runsOn": {
        "ref": "EmrClusterForBackup"
      },
      "type": "EmrActivity",
      "resizeClusterBeforeRunning": "false"
    },
    {
      "readThroughputPercent": "#{myDDBReadThroughputRatio}",
```

```
"name": "DDBSourceTable",
"id": "DDBSourceTable",
"type": "DynamoDBDataNode",
"tableName": "#{myDDBTableName}"

},
{
  "directoryPath": "#{myOutputS3Loc}/#{format(@scheduledStartTime, 'YYYY-MM-dd-HH-mm-ss')}",
  "name": "S3BackupLocation",
  "id": "S3BackupLocation",
  "type": "S3DataNode"
},
{
  "name": "EmrClusterForBackup",
  "coreInstanceCount": "1",
  "taskInstanceCount": "1",
  "taskInstanceType": "m4.xlarge",
  "coreInstanceType": "m4.xlarge",
  "releaseLabel": "emr-4.7.0",
  "masterInstanceType": "m4.xlarge",
  "id": "EmrClusterForBackup",
  "subnetId": "#{mySubnetId}",
  "emrManagedMasterSecurityGroupId": "#{myMasterSecurityGroup}",
  "emrManagedSlaveSecurityGroupId": "#{mySlaveSecurityGroup}",
  "serviceAccessSecurityGroupId": "#{myServiceAccessSecurityGroup}",
  "region": "#{myDDBRegion}",
  "type": "EmrCluster",
  "keyPair": "user-key-pair"
},
{
  "failureAndRerunMode": "CASCADE",
  "resourceRole": "DataPipelineDefaultResourceRole",
  "role": "DataPipelineDefaultRole",
  "pipelineLogUri": "#{myPipelineLogUri}",
  "scheduleType": "ONDEMAND",
  "name": "Default",
  "id": "Default"
}
],
"parameters": [
  {
    "description": "Output S3 folder",
    "id": "myOutputS3Loc",
    "type": "AWS::S3::ObjectKey"
  }
]
```

```
},
{
  "description": "Source DynamoDB table name",
  "id": "myDDBTableName",
  "type": "String"
},
{
  "default": "0.25",
  "watermark": "Enter value between 0.1-1.0",
  "description": "DynamoDB read throughput ratio",
  "id": "myDDBReadThroughputRatio",
  "type": "Double"
},
{
  "default": "us-east-1",
  "watermark": "us-east-1",
  "description": "Region of the DynamoDB table",
  "id": "myDDBRegion",
  "type": "String"
}
],
"values": {
  "myDDBRegion": "us-east-1",
  "myDDBTableName": "ddb_table",
  "myDDBReadThroughputRatio": "0.25",
  "myOutputS3Loc": "s3://s3_path",
  "mySubnetId": "subnet_id",
  "myServiceAccessSecurityGroup": "service access security group",
  "mySlaveSecurityGroup": "slave security group",
  "myMasterSecurityGroup": "master security group",
  "myPipelineLogUri": "s3://s3_path"
}
}
```

Attach EBS volumes to cluster nodes

Example

You can attach EBS volumes to any type of node in the EMR cluster within your pipeline. To attach EBS volumes to nodes, use `coreEbsConfiguration`, `masterEbsConfiguration`, and `TaskEbsConfiguration` in your `EmrCluster` configuration.

This example of the Amazon EMR cluster uses Amazon EBS volumes for its master, task, and core nodes. For more information, see [Amazon EBS volumes in Amazon EMR](#) in the *Amazon EMR Management Guide*.

These configurations are optional. You can use them in any pipeline that uses an `EmrCluster` object.

In the pipeline, click the `EmrCluster` object configuration, choose **Master EBS Configuration**, **Core EBS Configuration**, or **Task EBS Configuration**, and enter the configuration details similar to the following example.

```
{  
  "objects": [  
    {  
      "output": {  
        "ref": "S3BackupLocation"  
      },  
      "input": {  
        "ref": "DDBSourceTable"  
      },  
      "maximumRetries": "2",  
      "name": "TableBackupActivity",  
      "step": "s3://dynamodb-emr-#{myDDBRegion}/emr-ddb-storage-handler/2.1.0/emr-  
ddb-2.1.0.jar,org.apache.hadoop.dynamodb.tools.DynamoDbExport,#{output.directoryPath},#{input.t  
      "id": "TableBackupActivity",  
      "runsOn": {  
        "ref": "EmrClusterForBackup"  
      },  
      "type": "EmrActivity",  
      "resizeClusterBeforeRunning": "false"  
    },  
    {  
      "readThroughputPercent": "#{myDDBReadThroughputRatio}",  
      "name": "DDBSourceTable",  
      "id": "DDBSourceTable",  
      "type": "DynamoDBDataNode",  
      "tableName": "#{myDDBTableName}"  
    },  
    {  
      "directoryPath": "#{myOutputS3Loc}/#{format(@scheduledStartTime, 'YYYY-MM-dd-HH-  
mm-ss')}",  
      "name": "S3BackupLocation",  
      "id": "S3BackupLocation",  
    }  
  ]  
}
```

```
"type": "S3DataNode"
},
{
  "name": "EmrClusterForBackup",
  "coreInstanceCount": "1",
  "taskInstanceCount": "1",
  "taskInstanceType": "m4.xlarge",
  "coreInstanceType": "m4.xlarge",
  "releaseLabel": "emr-4.7.0",
  "masterInstanceType": "m4.xlarge",
  "id": "EmrClusterForBackup",
  "subnetId": "#{mySubnetId}",
  "emrManagedMasterSecurityGroupId": "#{myMasterSecurityGroup}",
  "emrManagedSlaveSecurityGroupId": "#{mySlaveSecurityGroup}",
  "region": "#{myDDBRegion}",
  "type": "EmrCluster",
  "coreEbsConfiguration": {
    "ref": "EBSSConfiguration"
  },
  "masterEbsConfiguration": {
    "ref": "EBSSConfiguration"
  },
  "taskEbsConfiguration": {
    "ref": "EBSSConfiguration"
  },
  "keyPair": "user-key-pair"
},
{
  "name": "EBSSConfiguration",
  "id": "EBSSConfiguration",
  "ebsOptimized": "true",
  "ebsBlockDeviceConfig" : [
    { "ref": "EbsBlockDeviceConfig" }
  ],
  "type": "EbsConfiguration"
},
{
  "name": "EbsBlockDeviceConfig",
  "id": "EbsBlockDeviceConfig",
  "type": "EbsBlockDeviceConfig",
  "volumesPerInstance" : "2",
  "volumeSpecification" : {
    "ref": "VolumeSpecification"
  }
}
```

```
},
{
  "name": "VolumeSpecification",
  "id": "VolumeSpecification",
  "type": "VolumeSpecification",
  "sizeInGB": "500",
  "volumeType": "io1",
  "iops": "1000"
},
{
  "failureAndRerunMode": "CASCADE",
  "resourceRole": "DataPipelineDefaultResourceRole",
  "role": "DataPipelineDefaultRole",
  "pipelineLogUri": "#{myPipelineLogUri}",
  "scheduleType": "ONDEMAND",
  "name": "Default",
  "id": "Default"
},
],
"parameters": [
  {
    "description": "Output S3 folder",
    "id": "myOutputs3Loc",
    "type": "AWS::S3::ObjectKey"
  },
  {
    "description": "Source DynamoDB table name",
    "id": "myDDBTableName",
    "type": "String"
  },
  {
    "default": "0.25",
    "watermark": "Enter value between 0.1-1.0",
    "description": "DynamoDB read throughput ratio",
    "id": "myDDBReadThroughputRatio",
    "type": "Double"
  },
  {
    "default": "us-east-1",
    "watermark": "us-east-1",
    "description": "Region of the DynamoDB table",
    "id": "myDDBRegion",
    "type": "String"
  }
]
```

```
],
"values": {
    "myDDBRegion": "us-east-1",
    "myDDBTableName": "ddb_table",
    "myDDBReadThroughputRatio": "0.25",
    "myOutputS3Loc": "s3://s3_path",
    "mySubnetId": "subnet_id",
    "mySlaveSecurityGroup": "slave security group",
    "myMasterSecurityGroup": "master security group",
    "myPipelineLogUri": "s3://s3_path"
}
}
```

See Also

- [EmrActivity](#)

HttpProxy

HttpProxy allows you to configure your own proxy and make Task Runner access the AWS Data Pipeline service through it. You do not need to configure a running Task Runner with this information.

Example of an HttpProxy in TaskRunner

The following pipeline definition shows an HttpProxy object:

```
{
  "objects": [
    {
      "schedule": {
        "ref": "Once"
      },
      "pipelineLogUri": "s3://myDPLogUri/path",
      "name": "Default",
      "id": "Default"
    },
    {
      "name": "test_proxy",
      "hostname": "hostname",
      "port": "port",
      "username": "username",
      "password": "password"
    }
  ]
}
```

```
    "password": "password",
    "windowsDomain": "windowsDomain",
    "type": "HttpProxy",
    "id": "test_proxy",
},
{
    "name": "ShellCommand",
    "id": "ShellCommand",
    "runsOn": {
        "ref": "Resource"
    },
    "type": "ShellCommandActivity",
    "command": "echo 'hello world' "
},
{
    "period": "1 day",
    "startDateTime": "2013-03-09T00:00:00",
    "name": "Once",
    "id": "Once",
    "endDateTime": "2013-03-10T00:00:00",
    "type": "Schedule"
},
{
    "role": "dataPipelineRole",
    "httpProxy": {
        "ref": "test_proxy"
    },
    "actionOnResourceFailure": "retrynone",
    "maximumRetries": "0",
    "type": "Ec2Resource",
    "terminateAfter": "10 minutes",
    "resourceRole": "resourceRole",
    "name": "Resource",
    "actionOnTaskFailure": "terminate",
    "securityGroups": "securityGroups",
    "keyPair": "keyPair",
    "id": "Resource",
    "region": "us-east-1"
}
],
"parameters": []
}
```

Syntax

Required Fields	Description	Slot Type
hostname	Host of the proxy which clients will use to connect to AWS Services.	String
port	Port of the proxy host which the clients will use to connect to AWS Services.	String

Optional Fields	Description	Slot Type
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref": "myBaseObjectId"}
*password	Password for proxy.	String
s3NoProxy	Disable the HTTP proxy when connecting to Amazon S3	Boolean
username	User name for proxy.	String
windowsDomain	The Windows domain name for NTLM Proxy.	String
windowsWorkgroup	The Windows workgroup name for NTLM Proxy.	String

Runtime Fields	Description	Slot Type
@version	Pipeline version the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object.	String
@pipelineId	Id of the pipeline to which this object belongs to.	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects.	String

Preconditions

The following are the AWS Data Pipeline precondition objects:

Objects

- [DynamoDBDataExists](#)
- [DynamoDBTableExists](#)
- [Exists](#)
- [S3KeyExists](#)
- [S3PrefixNotEmpty](#)
- [ShellCommandPrecondition](#)

DynamoDBDataExists

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

Syntax

Required Fields	Description	Slot Type
role	Specifies the role to be used to execute the precondition.	String

Required Fields	Description	Slot Type
tableName	DynamoDB Table to check.	String
Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
lateAfterTimeout	The elapsed time after pipeline start within which the object must complete. It is triggered only when the schedule type is not set to ondemand.	Period
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref": "myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref": "myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref": "myActionId"}

Optional Fields	Description	Slot Type
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref": "myBaseObjectId"}
preconditionTimeout	The period from start after which precondition is marked as failed if still not satisfied	Period
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set, then remote activities that do not report progress for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref": "myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of dependency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn":

Runtime Fields	Description	Slot Type
		{"ref":"myRunnable ObjectId"}
currentRetryCount	Number of times the precondition was tried in this attempt.	String
emrStepLog	EMR step logs available only on EMR activity attempts	String
errord	The errord if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
hostname	The host name of client that picked up the task attempt.	String
lastRetryTime	Last time when the precondition was tried within this attempt.	String
node	The node for which this precondition is being performed	Reference Object, e.g. "node":{"ref":"myRunnableObjectId"}
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object.	DateTime
@scheduledStartTime	Schedule start time for object.	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String

Runtime Fields	Description	Slot Type
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn":{"ref":"myRunnableObject Id"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object.	String
@pipelineId	Id of the pipeline to which this object belongs.	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects.	String

DynamoDBTableExists

A precondition to check that the DynamoDB table exists.

Syntax

Required Fields	Description	Slot Type
role	Specifies the role to be used to execute the precondition.	String
tableName	DynamoDB Table to check.	String

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
lateAfterTimeout	The elapsed time after pipeline start within which the object must complete. It is triggered only when the schedule type is not set to ondemand.	Period
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref": "myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref": "myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref": "myActionId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref": "myBaseObjectId"}
preconditionTimeout	The period from start after which precondition is marked as failed if still not satisfied	Period

Optional Fields	Description	Slot Type
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set, then remote activities that do not report progress for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnable ObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of dependency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnable ObjectId"}
currentRetryCount	Number of times the precondition was tried in this attempt.	String
emrStepLog	EMR step logs available only on EMR activity attempts	String

Runtime Fields	Description	Slot Type
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
hostname	The host name of client that picked up the task attempt.	String
lastRetryTime	Last time when the precondition was tried within this attempt.	String
node	The node for which this precondition is being performed	Reference Object, e.g. "node": {"ref": "myRunnableObjectId"}
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref": "myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

Exists

Checks whether a data node object exists.

 **Note**

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

Example

The following is an example of this object type. The InputData object references this object, Ready, plus another object that you'd define in the same pipeline definition file. CopyPeriod is a Schedule object.

```
{
  "id" : "InputData",
  "type" : "S3DataNode",
  "schedule" : { "ref" : "CopyPeriod" },
  "filePath" : "s3://amzn-s3-demo-bucket/InputData/#{@scheduledStartTime.format('YYYY-MM-dd-hh:mm')}.csv",
  "precondition" : { "ref" : "Ready" }
},
{
  "id" : "Ready",
}
```

```

    "type" : "Exists"
}

```

Syntax

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun.	Enumeration
lateAfterTimeout	The elapsed time after pipeline start within which the object must complete. It is triggered only when the schedule type is not set to ondemand.	Period
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref": "myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref": "myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref": "myActionId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref": "myBaseObjectId"}

Optional Fields	Description	Slot Type
preconditionTimeout	The period from start after which precondition is marked as failed if still not satisfied	Period
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set, then remote activities that do not report progress for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnable ObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of the dependency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnable ObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String

Runtime Fields	Description	Slot Type
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
hostname	The host name of client that picked up the task attempt.	String
node	The node for which this precondition is being performed.	Reference Object, e.g. "node":{"ref":"myRunnableObjectId"}
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object.	DateTime
@scheduledStartTime	Schedule start time for object.	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn":{"ref":"myRunnableObjectId"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object.	String

System Fields	Description	Slot Type
@pipelineId	Id of the pipeline to which this object belongs to.	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects.	String

See Also

- [ShellCommandPrecondition](#)

S3KeyExists

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

Example

The following is an example of this object type. The precondition will trigger when the key, s3://amzn-s3-demo-bucket/mykey, referenced by the s3Key parameter, exists.

```
{
  "id" : "InputReady",
  "type" : "S3KeyExists",
  "role" : "test-role",
  "s3Key" : "s3://amzn-s3-demo-bucket/mykey"
}
```

You can also use S3KeyExists as a precondition on the second pipeline that waits for the first pipeline to finish. To do so:

1. Write a file to Amazon S3 at the end of the first pipeline's completion.
2. Create an S3KeyExists precondition on the second pipeline.

Syntax

Required Fields	Description	Slot Type
role	Specifies the role to be used to execute the precondition.	String
s3Key	The Amazon S3 key.	String

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout before attempting to complete remote work one more time. If set, then a remote activity that does not complete within the set time after starting is attempted again.	Period
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun.	Enumeration
lateAfterTimeout	The elapsed time after pipeline start within which the object must complete. It is triggered only when the schedule type is not set to ondemand.	Period
maximumRetries	Maximum number of attempts that are initiated on failure.	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref": "myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref": "myActionId"}

Optional Fields	Description	Slot Type
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref": "myActionId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref": "myBaseObjectId"}
preconditionTimeout	The period from start after which precondition is marked as failed if still not satisfied.	Period
reportProgressTimeout	Timeout for remote work successive calls to <code>reportProgress</code> . If set, then remote activities that do not report progress for the specified period may be considered stalled and are retried.	Period
retryDelay	The timeout duration between two successive attempts.	Period

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref": "myRunnableObjectId"}
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String

Runtime Fields	Description	Slot Type
@cascadeFailedOn	Description of the dependency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnable ObjectId"}
currentRetryCount	Number of times the precondition was tried in this attempt.	String
emrStepLog	EMR step logs available only on EMR activity attempts	String
errord	The errord if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
hostname	The host name of client that picked up the task attempt.	String
lastRetryTime	Last time when the precondition was tried within this attempt.	String
node	The node for which this precondition is being performed	Reference Object, e.g. "node":{"ref":"myRunnableObjectId"}
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime
@scheduledStartTime	Schedule start time for object	DateTime

Runtime Fields	Description	Slot Type
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn":{"ref":"myRunnableObject Id"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

See Also

- [ShellCommandPrecondition](#)

S3PrefixNotEmpty

A precondition to check that the Amazon S3 objects with the given prefix (represented as a URI) are present.

Example

The following is an example of this object type using required, optional, and expression fields.

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

Syntax

Required Fields	Description	Slot Type
role	Specifies the role to be used to execute the precondition.	String
s3Prefix	The Amazon S3 prefix to check for existence of objects.	String
Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
lateAfterTimeout	The elapsed time after pipeline start within which the object must complete. It is triggered only when the schedule type is not set to ondemand.	Period
maximumRetries	Maximum number attempt retries on failure	Integer

Optional Fields	Description	Slot Type
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref": "myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref": "myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref": "myActionId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref": "myBaseObjectId"}
preconditionTimeout	The period from start after which precondition is marked as failed if still not satisfied	Period
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set, then remote activities that do not report progress for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref": "myRunnableObjectId"}

Runtime Fields	Description	Slot Type
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of the dependency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnableObjectId"}
currentRetryCount	Number of times the precondition was tried in this attempt.	String
emrStepLog	EMR step logs available only on EMR activity attempts	String
errord	The errord if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
hostname	The host name of client that picked up the task attempt.	String
lastRetryTime	Last time when the precondition was tried within this attempt.	String
node	The node for which this precondition is being performed.	Reference Object, e.g. "node": {"ref":"myRunnableObjectId"}

Runtime Fields	Description	Slot Type
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object.	DateTime
@scheduledStartTime	Schedule start time for object.	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref": "myRunnableObject", "id": "12345678901234567890123456789012"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

See Also

- [ShellCommandPrecondition](#)

ShellCommandPrecondition

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

Example

The following is an example of this object type.

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

Syntax

Required Group (One of the following is required)	Description	Slot Type
command	The command to run. This value and any associated parameters must function in the environment from which you are running the Task Runner.	String
scriptUri	An Amazon S3 URI path for a file to download and run as a shell command. Only one scriptUri or command field should be present. scriptUri cannot use parameters, use command instead.	String

Optional Fields	Description	Slot Type
attemptStatus	Most recently reported status from the remote activity.	String

Optional Fields	Description	Slot Type
attemptTimeout	Timeout for remote work completion. If set then a remote activity that does not complete within the set time of starting may be retried.	Period
failureAndRerunMode	Describes consumer node behavior when dependencies fail or are rerun	Enumeration
lateAfterTimeout	The elapsed time after pipeline start within which the object must complete. It is triggered only when the schedule type is not set to ondemand.	Period
maximumRetries	Maximum number attempt retries on failure	Integer
onFail	An action to run when current object fails.	Reference Object, e.g. "onFail": {"ref": "myActionId"}
onLateAction	Actions that should be triggered if an object has not yet been scheduled or still not completed.	Reference Object, e.g. "onLateAction": {"ref": "myActionId"}
onSuccess	An action to run when current object succeeds.	Reference Object, e.g. "onSuccess": {"ref": "myActionId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref": "myBaseObjectId"}
preconditionTimeout	The period from start after which precondition is marked as failed if still not satisfied	Period

Optional Fields	Description	Slot Type
reportProgressTimeout	Timeout for remote work successive calls to reportProgress. If set, then remote activities that do not report progress for the specified period may be considered stalled and so retried.	Period
retryDelay	The timeout duration between two retry attempts.	Period
scriptArgument	Argument to be passed to shell script	String
stderr	The Amazon S3 path that receives redirected system error messages from the command. If you use the runsOn field, this must be an Amazon S3 path because of the transitory nature of the resource running your activity. However, if you specify the workerGroup field, a local file path is permitted.	String
stdout	The Amazon S3 path that receives redirected output from the command. If you use the runsOn field, this must be an Amazon S3 path because of the transitory nature of the resource running your activity. However, if you specify the workerGroup field, a local file path is permitted.	String

Runtime Fields	Description	Slot Type
@activeInstances	List of the currently scheduled active instance objects.	Reference Object, e.g. "activeInstances": {"ref":"myRunnable ObjectId"}

Runtime Fields	Description	Slot Type
@actualEndTime	Time when the execution of this object finished.	DateTime
@actualStartTime	Time when the execution of this object started.	DateTime
cancellationReason	The cancellationReason if this object was cancelled.	String
@cascadeFailedOn	Description of the dependency chain the object failed on.	Reference Object, e.g. "cascadeFailedOn": {"ref":"myRunnable ObjectId"}
emrStepLog	EMR step logs available only on EMR activity attempts	String
errorId	The errorId if this object failed.	String
errorMessage	The errorMessage if this object failed.	String
errorStackTrace	The error stack trace if this object failed.	String
hadoopJobLog	Hadoop job logs available on attempts for EMR-based activities.	String
hostname	The host name of client that picked up the task attempt.	String
node	The node for which this precondition is being performed	Reference Object, e.g. "node": {"ref":"myRunnableObjectId"}
reportProgressTime	Most recent time that remote activity reported progress.	DateTime
@scheduledEndTime	Schedule end time for object	DateTime

Runtime Fields	Description	Slot Type
@scheduledStartTime	Schedule start time for object	DateTime
@status	The status of this object.	String
@version	Pipeline version the object was created with.	String
@waitingOn	Description of list of dependencies this object is waiting on.	Reference Object, e.g. "waitingOn": {"ref": "myRunnableObject Id"}

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

See Also

- [ShellCommandActivity](#)
- [Exists](#)

Databases

The following are the AWS Data Pipeline database objects:

Objects

- [JdbcDatabase](#)
- [RdsDatabase](#)
- [RedshiftDatabase](#)

JdbcDatabase

Defines a JDBC database.

Example

The following is an example of this object type.

```
{  
  "id" : "MyJdbcDatabase",  
  "type" : "JdbcDatabase",  
  "connectionString" : "jdbc:redshift://hostname:portnumber/dbname",  
  "jdbcDriverClass" : "com.amazon.redshift.jdbc41.Driver",  
  "jdbcDriverJarUri" : "s3://redshift-downloads/drivers/RedshiftJDBC41-1.1.6.1006.jar",  
  "username" : "user_name",  
  "*password" : "my_password"  
}
```

Syntax

Required Fields	Description	Slot Type
connectionString	The JDBC connection string to access the database.	String
jdbcDriverClass	The driver class to load before establishing the JDBC connection.	String
*password	The password to supply.	String
username	The user name to supply when connecting to the database.	String

Optional Fields	Description	Slot Type
databaseName	Name of the logical database to attach to	String
jdbcDriverJarUri	The location in Amazon S3 of the JDBC driver JAR file used to connect to the database. AWS Data Pipeline must have permission to read this JAR file.	String
jdbcProperties	Pairs of the form A=B that will be set as properties on JDBC connections for this database.	String
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref": "myBaseObjectId"}

Runtime Fields	Description	Slot Type
@version	Pipeline version that the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object.	String
@pipelineId	ID of the pipeline to which this object belongs.	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects.	String

RdsDatabase

Defines an Amazon RDS database.

Note

RdsDatabase does not support Aurora. Use [the section called "JdbcDatabase"](#) for Aurora, instead.

Example

The following is an example of this object type.

```
{  
    "id" : "MyRdsDatabase",  
    "type" : "RdsDatabase",  
    "region" : "us-east-1",  
    "username" : "user_name",  
    "*password" : "my_password",  
    "rdsInstanceId" : "my_db_instance_identifier"  
}
```

For the Oracle engine, the `jdbcDriverJarUri` field is required and you can specify the following driver: <http://www.oracle.com/technetwork/database/features/jdbc/jdbc-drivers-12c-download-1958347.html>. For the SQL Server engine, the `jdbcDriverJarUri` field is required and you can specify the following driver: <https://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=11774>. For the MySQL and PostgreSQL engines, the `jdbcDriverJarUri` field is optional.

Syntax

Required Fields	Description	Slot Type
*password	The password to supply.	String
rdsInstanceId	The DBInstanceIdentifier property of the DB instance.	String

Required Fields	Description	Slot Type
username	The user name to supply when connecting to the database.	String

Optional Fields	Description	Slot Type
databaseName	Name of the logical database to attach to	String
jdbcDriverJarUri	The location in Amazon S3 of the JDBC driver JAR file used to connect to the database. AWS Data Pipeline must have permission to read this JAR file. For the MySQL and PostgreSQL engines, the default driver is used if this field is not specified, but you can override the default using this field. For the Oracle and SQL Server engines, this field is required.	String
jdbcProperties	Pairs of the form A=B that will be set as properties on JDBC connections for this database.	String
parent	Parent of the current object from which slots will be inherited.	Reference Object, for example, "parent": {"ref": "myBaseObjectId"}
region	The code for the region where the database exists. For example, us-east-1.	String

Runtime Fields	Description	Slot Type
@version	Pipeline version that the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object.	String
@pipelineId	ID of the pipeline to which this object belongs.	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects.	String

RedshiftDatabase

Defines an Amazon Redshift database. RedshiftDatabase represents the properties of the database used by your pipeline.

Example

The following is an example of this object type.

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

By default, the object uses the Postgres driver, which requires the clusterId field. To use the Amazon Redshift driver, specify the Amazon Redshift database connection string from the Amazon Redshift console (starts with "jdbc:redshift:") in the connectionString field instead.

Syntax

Required Fields	Description	Slot Type
*password	The password to supply.	String

Required Fields	Description	Slot Type
username	The user name to supply when connecting to the database.	String

Required Group (One of the following is required)	Description	Slot Type
clusterId	The identifier provided by the user when the Amazon Redshift cluster was created. For example, if the endpoint for your Amazon Redshift cluster is mydb.example.us-east-1.redshift.amazonaws.com, the correct identifier is mydb. In the Amazon Redshift console, you can get this value from Cluster Identifier or Cluster Name.	String
connectionString	The JDBC endpoint for connecting to an Amazon Redshift instance owned by an account different than the pipeline. You can't specify both <code>connectionString</code> and <code>clusterId</code> .	String

Optional Fields	Description	Slot Type
databaseName	Name of the logical database to attach to.	String
jdbcProperties	Pairs of the form A=B to be set as properties on JDBC connections for this database.	String
parent	Parent of the current object from which slots are inherited.	Reference Object, for example, "parent":

Optional Fields	Description	Slot Type
		{"ref":"myBaseObjec tId"}
region	The code for the region where the database exists. For example, us-east-1.	Enumeration

Runtime Fields	Description	Slot Type
@version	Pipeline version that the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object.	String
@pipelineId	ID of the pipeline to which this object belongs.	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects.	String

Data Formats

The following are the AWS Data Pipeline data format objects:

Objects

- [CSV Data Format](#)
- [Custom Data Format](#)
- [DynamoDBDataFormat](#)
- [DynamoDBExportDataFormat](#)

- [RegEx Data Format](#)
- [TSV Data Format](#)

CSV Data Format

A comma-delimited data format where the column separator is a comma and the record separator is a newline character.

Example

The following is an example of this object type.

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

Syntax

Optional Fields	Description	Slot Type
column	Column name with datatype specified by each field for the data described by this data node. Ex: hostname STRING For multiple values, use column names and data types separated by a space.	String
escapeChar	A character, for example "\", that instructs the parser to ignore the next character.	String
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref": "myBaseObjectId"}

Runtime Fields	Description	Slot Type
@version	Pipeline version the object was created with.	String
System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

Custom Data Format

A custom data format defined by a combination of a certain column separator, record separator, and escape character.

Example

The following is an example of this object type.

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

Syntax

Required Fields	Description	Slot Type
columnSeparator	A character that indicates the end of a column in a data file.	String

Optional Fields	Description	Slot Type
column	Column name with datatype specified by each field for the data described by this data node. Ex: hostname STRING For multiple values, use column names and data types separated by a space.	String
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref": "myBaseObjectIId"}
recordSeparator	A character that indicates the end of a row in a data file, for example "\n". Only single characters are supported.	String

Runtime Fields	Description	Slot Type
@version	Pipeline version the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String

System Fields	Description	Slot Type
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

DynamoDBDataFormat

Applies a schema to a DynamoDB table to make it accessible by a Hive query.

DynamoDBDataFormat is used with a `HiveActivity` object and a `DynamoDBDataNode` input and output. DynamoDBDataFormat requires that you specify all columns in your Hive query. For more flexibility to specify certain columns in a Hive query or Amazon S3 support, see [DynamoDBExportDataFormat](#).

 **Note**

DynamoDB Boolean types are not mapped to Hive Boolean types. However, it is possible to map DynamoDB integer values of 0 or 1 to Hive Boolean types.

Example

The following example shows how to use `DynamoDBDataFormat` to assign a schema to a `DynamoDBDataNode` input, which allows a `HiveActivity` object to access the data by named columns and copy the data to a `DynamoDBDataNode` output.

```
{
  "objects": [
    {
      "id" : "Exists.1",
      "name" : "Exists.1",
      "type" : "Exists"
    },
    {
      "id" : "DataFormat.1",
      "name" : "DataFormat.1",
      "type" : "DataFormat"
    }
  ]
}
```

```
"name" : "DataFormat.1",
"type" : "DynamoDBDataFormat",
"column" : [
    "hash STRING",
    "range STRING"
],
},
{
    "id" : "DynamoDBDataNode.1",
    "name" : "DynamoDBDataNode.1",
    "type" : "DynamoDBDataNode",
    "tableName" : "$INPUT_TABLE_NAME",
    "schedule" : { "ref" : "ResourcePeriod" },
    "dataFormat" : { "ref" : "DataFormat.1" }
},
{
    "id" : "DynamoDBDataNode.2",
    "name" : "DynamoDBDataNode.2",
    "type" : "DynamoDBDataNode",
    "tableName" : "$OUTPUT_TABLE_NAME",
    "schedule" : { "ref" : "ResourcePeriod" },
    "dataFormat" : { "ref" : "DataFormat.1" }
},
{
    "id" : "EmrCluster.1",
    "name" : "EmrCluster.1",
    "type" : "EmrCluster",
    "schedule" : { "ref" : "ResourcePeriod" },
    "masterInstanceType" : "m1.small",
    "keyPair" : "$KEYPAIR"
},
{
    "id" : "HiveActivity.1",
    "name" : "HiveActivity.1",
    "type" : "HiveActivity",
    "input" : { "ref" : "DynamoDBDataNode.1" },
    "output" : { "ref" : "DynamoDBDataNode.2" },
    "schedule" : { "ref" : "ResourcePeriod" },
    "runsOn" : { "ref" : "EmrCluster.1" },
    "hiveScript" : "insert overwrite table ${output1} select * from ${input1} ;"
},
{
    "id" : "ResourcePeriod",
    "name" : "ResourcePeriod",
```

```

    "type" : "Schedule",
    "period" : "1 day",
    "startDateTime" : "2012-05-04T00:00:00",
    "endDateTime" : "2012-05-05T00:00:00"
}
]
}

```

Syntax

Optional Fields	Description	Slot Type
column	The column name with data type specified by each field for the data described by this data node. For example, hostname STRING. For multiple values, use column names and data types separated by a space.	String
parent	The parent of the current object from which slots will be inherited.	Reference Object, such as "parent": {"ref":"myBaseObjectId"}

Runtime Fields	Description	Slot Type
@version	The pipeline version uses to create the object.	String

System Fields	Description	Slot Type
@error	The error describing the ill-formed object.	String
@pipelineId	The Id of the pipeline to which this object belongs.	String

System Fields	Description	Slot Type
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects.	String

DynamoDBExportDataFormat

Applies a schema to an DynamoDB table to make it accessible by a Hive query. Use `DynamoDBExportDataFormat` with a `HiveCopyActivity` object and `DynamoDBDataNode` or `S3DataNode` input and output. `DynamoDBExportDataFormat` has the following benefits:

- Provides both DynamoDB and Amazon S3 support
- Allows you to filter data by certain columns in your Hive query
- Exports all attributes from DynamoDB even if you have a sparse schema

 **Note**

DynamoDB Boolean types are not mapped to Hive Boolean types. However, it is possible to map DynamoDB integer values of 0 or 1 to Hive Boolean types.

Example

The following example shows how to use `HiveCopyActivity` and `DynamoDBExportDataFormat` to copy data from one `DynamoDBDataNode` to another, while filtering based on a time stamp.

```
{
  "objects": [
    {
      "id" : "DataFormat.1",
      "name" : "DataFormat.1",
      "type" : "DynamoDBExportDataFormat",
      "column" : "timeStamp BIGINT"
    },
  ]
}
```

```
{  
    "id" : "DataFormat.2",  
    "name" : "DataFormat.2",  
    "type" : "DynamoDBExportDataFormat"  
,  
{  
    "id" : "DynamoDBDataNode.1",  
    "name" : "DynamoDBDataNode.1",  
    "type" : "DynamoDBDataNode",  
    "tableName" : "item_mapped_table_restore_temp",  
    "schedule" : { "ref" : "ResourcePeriod" },  
    "dataFormat" : { "ref" : "DataFormat.1" }  
,  
{  
    "id" : "DynamoDBDataNode.2",  
    "name" : "DynamoDBDataNode.2",  
    "type" : "DynamoDBDataNode",  
    "tableName" : "restore_table",  
    "region" : "us_west_1",  
    "schedule" : { "ref" : "ResourcePeriod" },  
    "dataFormat" : { "ref" : "DataFormat.2" }  
,  
{  
    "id" : "EmrCluster.1",  
    "name" : "EmrCluster.1",  
    "type" : "EmrCluster",  
    "schedule" : { "ref" : "ResourcePeriod" },  
    "masterInstanceType" : "m1.xlarge",  
    "coreInstanceCount" : "4"  
,  
{  
    "id" : "HiveTransform.1",  
    "name" : "Hive Copy Transform.1",  
    "type" : "HiveCopyActivity",  
    "input" : { "ref" : "DynamoDBDataNode.1" },  
    "output" : { "ref" : "DynamoDBDataNode.2" },  
    "schedule" : { "ref" : "ResourcePeriod" },  
    "runsOn" : { "ref" : "EmrCluster.1" },  
    "filterSql" : " `timeStamp` > unix_timestamp(\"#{@scheduledStartTime}\", \"yyyy-MM-dd'T'HH:mm:ss\")"  
,  
{  
    "id" : "ResourcePeriod",  
    "name" : "ResourcePeriod",  
}
```

```

    "type" : "Schedule",
    "period" : "1 Hour",
    "startDateTime" : "2013-06-04T00:00:00",
    "endDateTime" : "2013-06-04T01:00:00"
}
]
}

```

Syntax

Optional Fields	Description	Slot Type
column	Column name with datatype specified by each field for the data described by this data node. Ex: hostname STRING	String
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref": "myBaseObjectId"}

Runtime Fields	Description	Slot Type
@version	Pipeline version the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

RegEx Data Format

A custom data format defined by a regular expression.

Example

The following is an example of this object type.

```
{
  "id" : "MyInputDataType",
  "type" : "RegEx",
  "inputRegEx" : "([^\n]*)([^^\n]*)([^^\n]*)(-|\\[[^\n\\]]*\\]) ([^\n]*|[^\n]*\\") (-|[0-9]*)(-[0-9]*)(?: ([^\n]*|[^\n]*\\") ([^\n]*|[^\n]*\\"))?",
  "outputFormat" : "%1$s %2$s %3$s %4$s %5$s %6$s %7$s %8$s %9$s",
  "column" : [
    "host STRING",
    "identity STRING",
    "user STRING",
    "time STRING",
    "request STRING",
    "status STRING",
    "size STRING",
    "referer STRING",
    "agent STRING"
  ]
}
```

Syntax

Optional Fields	Description	Slot Type
column	Column name with datatype specified by each field for the data described by this data node. Ex: hostname STRING For multiple values, use column names and data types separated by a space.	String
inputRegEx	The regular expression to parse an S3 input file. inputRegEx provides a way to retrieve	String

Optional Fields	Description	Slot Type
	columns from relatively unstructured data in a file.	
outputFormat	The column fields retrieved by inputRegEx, but referenced as %1\$s %2\$s using Java formatter syntax.	String
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent":{"ref":"myBaseObjectId"}

Runtime Fields	Description	Slot Type
@version	Pipeline version the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

TSV Data Format

A comma-delimited data format where the column separator is a tab character and the record separator is a newline character.

Example

The following is an example of this object type.

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

Syntax

Optional Fields	Description	Slot Type
column	Column name and data type for the data described by this data node. For example "Name STRING" denotes a column named Name with fields of data type STRING. Separate multiple column name and data type pairs with commas (as shown in the example).	String
columnSeparator	The character that separates fields in one column from fields in the next column. Defaults to '\t'.	String
escapeChar	A character, for example "\", that instructs the parser to ignore the next character.	String
parent	Parent of the current object from which slots are inherited.	Reference Object, for example, "parent": {"ref":"myBaseObjectId"}
recordSeparator	The character that separates records. Defaults to '\n'.	String

Runtime Fields	Description	Slot Type
@version	Pipeline version that the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object.	String
@pipelineId	ID of the pipeline to which this object belongs.	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects, which execute Attempt Objects.	String

Actions

The following are the AWS Data Pipeline action objects:

Objects

- [SnsAlarm](#)
- [Terminate](#)

SnsAlarm

Sends an Amazon SNS notification message when an activity fails or finishes successfully.

Example

The following is an example of this object type. The values for node.input and node.output come from the data node or activity that references this object in its onSuccess field.

```
{
  "id" : "SuccessNotify",
```

```

"name" : "SuccessNotify",
"type" : "SnsAlarm",
"topicArn" : "arn:aws:sns:us-east-1:28619EXAMPLE:ExampleTopic",
"subject" : "COPY SUCCESS: #{node.@scheduledStartTime}",
"message" : "Files were copied from #{node.input} to #{node.output}."
}

```

Syntax

Required Fields	Description	Slot Type
message	The body text of the Amazon SNS notification.	String
role	The IAM role to use to create the Amazon SNS alarm.	String
subject	The subject line of the Amazon SNS notification message.	String
topicArn	The destination Amazon SNS topic ARN for the message.	String

Optional Fields	Description	Slot Type
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref": "myBaseObjectId"}

Runtime Fields	Description	Slot Type
node	The node for which this action is being performed.	Reference Object, e.g. "node": {"ref": "myUnnableObjectId"}
@version	Pipeline version the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object.	String
@pipelineId	Id of the pipeline to which this object belongs to.	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects.	String

Terminate

An action to trigger the cancellation of a pending or unfinished activity, resource, or data node. AWS Data Pipeline attempts to put the activity, resource, or data node into the CANCELLED state if it does not start by the lateAfterTimeout value.

You cannot terminate actions that include onSuccess, OnFail, or onLateAction resources.

Example

The following is an example of this object type. In this example, the onLateAction field of MyActivity contains a reference to the action DefaultAction1. When you provide an action for onLateAction, you must also provide a lateAfterTimeout value to indicate the period of time since the scheduled start of the pipeline after which the activity is considered late.

```
{
  "name" : "MyActivity",
  "id" : "DefaultActivity1",
  "schedule" : {
    "ref" : "MySchedule"
  },
  "runsOn" : {
    "ref" : "MyEmrCluster"
  },
  "lateAfterTimeout" : "1 Hours",
  "type" : "EmrActivity",
  "onLateAction" : {
    "action" : "DefaultAction1"
  }
}
```

```

    "ref" : "DefaultAction1"
},
"step" : [
    "s3://amzn-s3-demo-bucket/myPath/myStep.jar,firstArg,secondArg",
    "s3://amzn-s3-demo-bucket/myPath/myOtherStep.jar,anotherArg"
]
},
{
    "name" : "TerminateTasks",
    "id" : "DefaultAction1",
    "type" : "Terminate"
}

```

Syntax

Optional Fields	Description	Slot Type
parent	Parent of the current object from which slots are inherited.	Reference Object, for example "parent": {"ref":"myBaseObjectId"}

Runtime Fields	Description	Slot Type
node	The node for which this action is being performed.	Reference Object, for example "node": {"ref":"myRunnableObject"}
@version	Pipeline version that the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object.	String

System Fields	Description	Slot Type
@pipelineId	ID of the pipeline to which this object belongs.	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects, which execute Attempt Objects.	String

Schedule

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

Note

When a schedule's start time is in the past, AWS Data Pipeline backfills your pipeline and begins scheduling runs immediately beginning at the specified start time. For testing/development, use a relatively short interval. Otherwise, AWS Data Pipeline attempts to queue and schedule all runs of your pipeline for that interval. AWS Data Pipeline attempts to prevent accidental backfills if the pipeline component scheduledStartTime is earlier than 1 day ago by blocking pipeline activation.

Examples

The following is an example of this object type. It defines a schedule of every hour starting at 00:00:00 hours on 2012-09-01 and ending at 00:00:00 hours on 2012-10-01. The first period ends at 01:00:00 on 2012-09-01.

```
{  
  "id" : "Hourly",  
  "type" : "Schedule",  
  "period" : "1 hours",  
  "startDateTime" : "2012-09-01T00:00:00",  
  "endDateTime" : "2012-10-01T00:00:00"  
}
```

The following pipeline will start at the FIRST_ACTIVATION_DATE_TIME and run every hour until 22:00:00 hours on 2014-04-25.

```
{  
    "id": "SchedulePeriod",  
    "name": "SchedulePeriod",  
    "startAt": "FIRST_ACTIVATION_DATE_TIME",  
    "period": "1 hours",  
    "type": "Schedule",  
    "endDateTime": "2014-04-25T22:00:00"  
}
```

The following pipeline will start at the FIRST_ACTIVATION_DATE_TIME and run every hour and complete after three occurrences.

```
{  
    "id": "SchedulePeriod",  
    "name": "SchedulePeriod",  
    "startAt": "FIRST_ACTIVATION_DATE_TIME",  
    "period": "1 hours",  
    "type": "Schedule",  
    "occurrences": "3"  
}
```

The following pipeline will start at 22:00:00 on 2014-04-25, run hourly, and end after three occurrences.

```
{  
    "id": "SchedulePeriod",  
    "name": "SchedulePeriod",  
    "startDateTime": "2014-04-25T22:00:00",  
    "period": "1 hours",  
    "type": "Schedule",  
    "occurrences": "3"  
}
```

On-demand using the Default object

```
{  
    "name": "Default",  
    "resourceRole": "DataPipelineDefaultResourceRole",  
}
```

```
"role": "DataPipelineDefaultRole",
"scheduleType": "ondemand"
}
```

On-demand with explicit Schedule object

```
{
  "name": "Default",
  "resourceRole": "DataPipelineDefaultResourceRole",
  "role": "DataPipelineDefaultRole",
  "scheduleType": "ondemand"
},
{
  "name": "DefaultSchedule",
  "type": "Schedule",
  "id": "DefaultSchedule",
  "period": "ONDEMAND_PERIOD",
  "startAt": "ONDEMAND_ACTIVATION_TIME"
},
```

The following examples demonstrate how a Schedule can be inherited from the default object, be explicitly set for that object, or be given by a parent reference:

Schedule inherited from Default object

```
{
  "objects": [
    {
      "id": "Default",
      "failureAndRerunMode": "cascade",
      "resourceRole": "DataPipelineDefaultResourceRole",
      "role": "DataPipelineDefaultRole",
      "pipelineLogUri": "s3://myLogsbucket",
      "scheduleType": "cron",
      "schedule": {
        "ref": "DefaultSchedule"
      }
    },
    {
      "type": "Schedule",
      "id": "DefaultSchedule",
      "occurrences": "1",
      "period": "1 Day",
    }
  ]
}
```

```
"startAt": "FIRST_ACTIVATION_DATE_TIME"
},
{
  "id": "A_Fresh_NewEC2Instance",
  "type": "Ec2Resource",
  "terminateAfter": "1 Hour"
},
{
  "id": "ShellCommandActivity_HelloWorld",
  "runsOn": {
    "ref": "A_Fresh_NewEC2Instance"
  },
  "type": "ShellCommandActivity",
  "command": "echo 'Hello World!''"
}
]
```

Explicit schedule on the object

```
{
  "objects": [
    {
      "id": "Default",
      "failureAndRerunMode": "cascade",
      "resourceRole": "DataPipelineDefaultResourceRole",
      "role": "DataPipelineDefaultRole",
      "pipelineLogUri": "s3://myLogsbucket",
      "scheduleType": "cron"

    },
    {
      "type": "Schedule",
      "id": "DefaultSchedule",
      "occurrences": "1",
      "period": "1 Day",
      "startAt": "FIRST_ACTIVATION_DATE_TIME"
    },
    {
      "id": "A_Fresh_NewEC2Instance",
      "type": "Ec2Resource",
      "terminateAfter": "1 Hour"
    }
  ]
}
```

```
{  
    "id": "ShellCommandActivity_HelloWorld",  
    "runsOn": {  
        "ref": "A_Fresh_NewEC2Instance"  
    },  
    "schedule": {  
        "ref": "DefaultSchedule"  
    },  
    "type": "ShellCommandActivity",  
    "command": "echo 'Hello World!'"  
}  
]  
}
```

Schedule from Parent reference

```
{  
    "objects": [  
        {  
            "id": "Default",  
            "failureAndRerunMode": "cascade",  
            "resourceRole": "DataPipelineDefaultResourceRole",  
            "role": "DataPipelineDefaultRole",  
            "pipelineLogUri": "s3://myLogsbucket",  
            "scheduleType": "cron"  
        },  
        {  
            "id": "parent1",  
            "schedule": {  
                "ref": "DefaultSchedule"  
            }  
        },  
        {  
            "type": "Schedule",  
            "id": "DefaultSchedule",  
            "occurrences": "1",  
            "period": "1 Day",  
            "startAt": "FIRST_ACTIVATION_DATE_TIME"  
        },  
        {  
            "id": "A_Fresh_NewEC2Instance",  
            "runsOn": {  
                "ref": "parent1"  
            }  
        }  
    ]  
}
```

```

    "type": "Ec2Resource",
    "terminateAfter": "1 Hour"
},
{
  "id": "ShellCommandActivity_HelloWorld",
  "runsOn": {
    "ref": "A_Fresh_NewEC2Instance"
  },
  "parent": {
    "ref": "parent1"
  },
  "type": "ShellCommandActivity",
  "command": "echo 'Hello World!''"
}
]
}

```

Syntax

Required Fields	Description	Slot Type
period	How often the pipeline should run. The format is "N [minutes hours days weeks months]", where N is a number followed by one of the time specifiers. For example, "15 minutes", runs the pipeline every 15 minutes. The minimum period is 15 minutes and the maximum period is 3 years.	Period
Required Group (One of the following is required)	Description	Slot Type
startAt	The date and time at which to start the scheduled pipeline runs. Valid value is FIRST_ACTIVATION_DATE_TIME, which is	Enumeration

Required Group (One of the following is required)	Description	Slot Type
	deprecated in favor of creating an on-demand pipeline.	
startDateTime	The date and time to start the scheduled runs. You must use either startDateTime or startAt but not both.	DateTime

Optional Fields	Description	Slot Type
endDateTime	The date and time to end the scheduled runs. Must be a date and time later than the value of startDateTime or startAt. The default behavior is to schedule runs until the pipeline is shut down.	DateTime
occurrences	The number of times to execute the pipeline after it's activated. You can't use occurrences with endDateTime.	Integer
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent": {"ref": "myBaseObjectIId"}

Runtime Fields	Description	Slot Type
@version	Pipeline version the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@firstActivationTime	The time of object creation.	DateTime
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

Utilities

The following utility objects configure other pipeline objects:

Topics

- [ShellScriptConfig](#)
- [EmrConfiguration](#)
- [Property](#)

ShellScriptConfig

Use with an Activity to run a shell script for preActivityTaskConfig and postActivityTaskConfig. This object is available for [HadoopActivity](#), [HiveActivity](#), [HiveCopyActivity](#), and [PigActivity](#). You specify an S3 URI and a list of arguments for the script.

Example

A ShellScriptConfig with arguments:

```
{
  "id" : "ShellScriptConfig_1",
  "name" : "prescript",
  "type" : "ShellScriptConfig",
```

```

"scriptUri": "s3://my-bucket/shell-cleanup.sh",
"scriptArgument" : ["arg1","arg2"]
}

```

Syntax

This object includes the following fields.

Optional Fields	Description	Slot Type
parent	Parent of the current object from which slots are inherited.	Reference Object, for example, "parent": {"ref":"myBaseObjectId"}
scriptArgument	A list of arguments to use with the shell script.	String
scriptUri	The script URI in Amazon S3 that should be downloaded and run.	String

Runtime Fields	Description	Slot Type
@version	Pipeline version that the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object.	String
@pipelineId	ID of the pipeline to which this object belongs.	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects, which execute Attempt Objects.	String

EmrConfiguration

The EmrConfiguration object is the configuration used for EMR clusters with releases 4.0.0 or greater. Configurations (as a list) is a parameter to the RunJobFlow API call. The configuration API for Amazon EMR takes a classification and properties. AWS Data Pipeline uses EmrConfiguration with corresponding Property objects to configure an [EmrCluster](#) application such as Hadoop, Hive, Spark, or Pig on EMR clusters launched in a pipeline execution. Because configuration can only be changed for new clusters, you cannot provide a EmrConfiguration object for existing resources. For more information, see <http://docs.aws.amazon.com/ElasticMapReduce/latest/ReleaseGuide/>.

Example

The following configuration object sets the `io.file.buffer.size` and `fs.s3.block.size` properties in `core-site.xml`:

```
[  
  {  
    "classification": "core-site",  
    "properties":  
    {  
      "io.file.buffer.size": "4096",  
      "fs.s3.block.size": "67108864"  
    }  
  }  
]
```

The corresponding pipeline object definition uses a EmrConfiguration object and a list of Property objects in the property field:

```
{  
  "objects": [  
    {  
      "name": "ReleaseLabelCluster",  
      "releaseLabel": "emr-4.1.0",  
      "applications": ["spark", "hive", "pig"],  
      "id": "ResourceId_I1mCc",  
      "type": "EmrCluster",  
      "configuration": {  
        "ref": "coresite"  
      }  
    },  
  ]  
}
```

```
{  
    "name": "coresite",  
    "id": "coresite",  
    "type": "EmrConfiguration",  
    "classification": "core-site",  
    "property": [{  
        "ref": "io-file-buffer-size"  
    },  
    {  
        "ref": "fs-s3-block-size"  
    }  
]  
,  
{  
    "name": "io-file-buffer-size",  
    "id": "io-file-buffer-size",  
    "type": "Property",  
    "key": "io.file.buffer.size",  
    "value": "4096"  
,  
{  
    "name": "fs-s3-block-size",  
    "id": "fs-s3-block-size",  
    "type": "Property",  
    "key": "fs.s3.block.size",  
    "value": "67108864"  
}  
]  
}
```

The following example is a nested configuration used to set the Hadoop environment with the `hadoop-env` classification:

```
[  
{  
    "classification": "hadoop-env",  
    "properties": {},  
    "configurations": [  
        {  
            "classification": "export",  
            "properties": {  
                "YARN_PROXYSERVER_HEAPSIZE": "2396"  
            }  
    ]  
}
```

```
    }
]
}
]
```

The corresponding pipeline definition object that uses this configuration is below:

```
{
  "objects": [
    {
      "name": "ReleaseLabelCluster",
      "releaseLabel": "emr-4.0.0",
      "applications": ["spark", "hive", "pig"],
      "id": "ResourceId_I1mCc",
      "type": "EmrCluster",
      "configuration": {
        "ref": "hadoop-env"
      }
    },
    {
      "name": "hadoop-env",
      "id": "hadoop-env",
      "type": "EmrConfiguration",
      "classification": "hadoop-env",
      "configuration": {
        "ref": "export"
      }
    },
    {
      "name": "export",
      "id": "export",
      "type": "EmrConfiguration",
      "classification": "export",
      "property": {
        "ref": "yarn-proxyserver-heapsize"
      }
    },
    {
      "name": "yarn-proxyserver-heapsize",
      "id": "yarn-proxyserver-heapsize",
      "type": "Property",
      "key": "YARN_PROXYSERVER_HEAPSIZE",
      "value": "2396"
    }
  ]
}
```

```
    },
]
}
```

The following example modifies a Hive-specific property for an EMR cluster:

```
{
  "objects": [
    {
      "name": "hivesite",
      "id": "hivesite",
      "type": "EmrConfiguration",
      "classification": "hive-site",
      "property": [
        {
          "ref": "hive-client-timeout"
        }
      ]
    },
    {
      "name": "hive-client-timeout",
      "id": "hive-client-timeout",
      "type": "Property",
      "key": "hive.metastore.client.socket.timeout",
      "value": "2400s"
    }
  ]
}
```

Syntax

This object includes the following fields.

Required Fields	Description	Slot Type
classification	Classification for the configuration.	String

Optional Fields	Description	Slot Type
configuration	Sub-configuration for this configuration.	Reference Object, e.g. "configuration":{ "ref":"myEmrConfigurationId"}
parent	Parent of the current object from which slots will be inherited.	Reference Object, e.g. "parent":{ "ref": "myBaseObjectId"}
property	Configuration property.	Reference Object, e.g. "property":{ "ref": "myPropertyId"}

Runtime Fields	Description	Slot Type
@version	Pipeline version the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object	String
@pipelineId	Id of the pipeline to which this object belongs to	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects which execute Attempt Objects	String

See Also

- [EmrCluster](#)

- [Property](#)
- [Amazon EMR Release Guide](#)

Property

A single key-value property for use with an EmrConfiguration object.

Example

The following pipeline definition shows an EmrConfiguration object and corresponding Property objects to launch an EmrCluster:

```
{  
  "objects": [  
    {  
      "name": "ReleaseLabelCluster",  
      "releaseLabel": "emr-4.1.0",  
      "applications": ["spark", "hive", "pig"],  
      "id": "ResourceId_I1mCc",  
      "type": "EmrCluster",  
      "configuration": {  
        "ref": "coresite"  
      }  
    },  
    {  
      "name": "coresite",  
      "id": "coresite",  
      "type": "EmrConfiguration",  
      "classification": "core-site",  
      "property": [{  
        "ref": "io-file-buffer-size"  
      },  
      {  
        "ref": "fs-s3-block-size"  
      }  
    ],  
    {  
      "name": "io-file-buffer-size",  
      "id": "io-file-buffer-size",  
      "type": "Property",  
      "key": "io.file.buffer.size",  
      "value": "134217728"  
    }  
  ]  
}
```

```

    "value": "4096"
},
{
  "name": "fs-s3-block-size",
  "id": "fs-s3-block-size",
  "type": "Property",
  "key": "fs.s3.block.size",
  "value": "67108864"
}
]
}

```

Syntax

This object includes the following fields.

Required Fields	Description	Slot Type
key	key	String
value	value	String

Optional Fields	Description	Slot Type
parent	Parent of the current object from which slots are inherited.	Reference Object, for example, "parent": {"ref":"myBaseObjId"}

Runtime Fields	Description	Slot Type
@version	Pipeline version that the object was created with.	String

System Fields	Description	Slot Type
@error	Error describing the ill-formed object.	String
@pipelineId	ID of the pipeline to which this object belongs.	String
@sphere	The sphere of an object denotes its place in the lifecycle: Component Objects give rise to Instance Objects, which execute Attempt Objects.	String

See Also

- [EmrCluster](#)
- [EmrConfiguration](#)
- [Amazon EMR Release Guide](#)

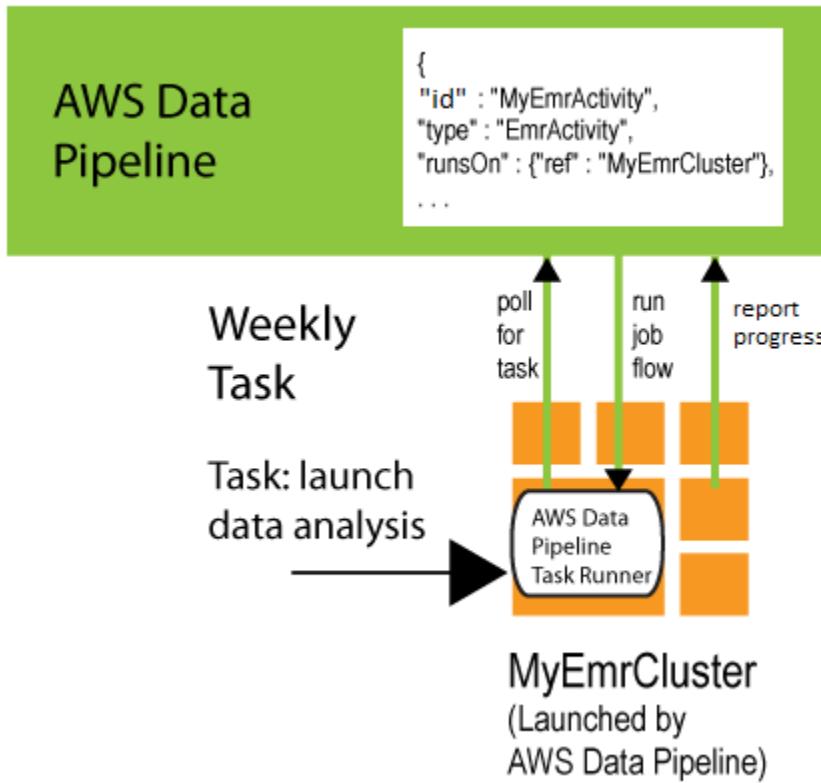
Working with Task Runner

Task Runner is a task agent application that polls AWS Data Pipeline for scheduled tasks and executes them on Amazon EC2 instances, Amazon EMR clusters, or other computational resources, reporting status as it does so. Depending on your application, you may choose to:

- Allow AWS Data Pipeline to install and manage one or more Task Runner applications for you. When a pipeline is activated, the default `Ec2Instance` or `EmrCluster` object referenced by an activity `runsOn` field is automatically created. AWS Data Pipeline takes care of installing Task Runner on an EC2 instance or on the master node of an EMR cluster. In this pattern, AWS Data Pipeline can do most of the instance or cluster management for you.
- Run all or parts of a pipeline on resources that you manage. The potential resources include a long-running Amazon EC2 instance, an Amazon EMR cluster, or a physical server. You can install a task runner (which can be either Task Runner or a custom task agent of your own devise) almost anywhere, provided that it can communicate with the AWS Data Pipeline web service. In this pattern, you assume almost complete control over which resources are used and how they are managed, and you must manually install and configure Task Runner. To do so, use the procedures in this section, as described in [Executing Work on Existing Resources Using Task Runner](#).

Task Runner on AWS Data Pipeline-Managed Resources

When a resource is launched and managed by AWS Data Pipeline, the web service automatically installs Task Runner on that resource to process tasks in the pipeline. You specify a computational resource (either an Amazon EC2 instance or an Amazon EMR cluster) for the `runsOn` field of an activity object. When AWS Data Pipeline launches this resource, it installs Task Runner on that resource and configures it to process all activity objects that have their `runsOn` field set to that resource. When AWS Data Pipeline terminates the resource, the Task Runner logs are published to an Amazon S3 location before it shuts down.



For example, if you use the `EmrActivity` in a pipeline, and specify an `EmrCluster` resource in the `runsOn` field. When AWS Data Pipeline processes that activity, it launches an Amazon EMR cluster and installs Task Runner onto the master node. This Task Runner then processes the tasks for activities that have their `runsOn` field set to that `EmrCluster` object. The following excerpt from a pipeline definition shows this relationship between the two objects.

```
{  
  "id" : "MyEmrActivity",  
  "name" : "Work to perform on my data",  
  "type" : "EmrActivity",  
  "runsOn" : {"ref" : "MyEmrCluster"},  
  "preStepCommand" : "scp remoteFiles localFiles",  
  "step" : "s3://amzn-s3-demo-bucket/myPath/myStep.jar,firstArg,secondArg",  
  "step" : "s3://amzn-s3-demo-bucket/myPath/myOtherStep.jar,anotherArg",  
  "postStepCommand" : "scp localFiles remoteFiles",  
  "input" : {"ref" : "MyS3Input"},  
  "output" : {"ref" : "MyS3Output"}  
},  
{  
  "id" : "MyEmrCluster",  
  "name" : "EMR cluster to perform the work",  
  ...  
}
```

```
"type" : "EmrCluster",
".hadoopVersion" : "0.20",
"keypair" : "myKeyPair",
"masterInstanceType" : "m1.xlarge",
"coreInstanceType" : "m1.small",
"coreInstanceCount" : "10",
"taskInstanceType" : "m1.small",
"taskInstanceCount": "10",
"bootstrapAction" : "s3://elasticmapreduce/libs/ba/configure-hadoop,arg1,arg2,arg3",
"bootstrapAction" : "s3://elasticmapreduce/libs/ba/configure-other-stuff,arg1,arg2"
}
```

For information and examples of running this activity, see [EmrActivity](#).

If you have multiple AWS Data Pipeline-managed resources in a pipeline, Task Runner is installed on each of them, and they all poll AWS Data Pipeline for tasks to process.

Executing Work on Existing Resources Using Task Runner

You can install Task Runner on computational resources that you manage, such as an Amazon EC2 instance, or a physical server or workstation. Task Runner can be installed anywhere, on any compatible hardware or operating system, provided that it can communicate with the AWS Data Pipeline web service.

This approach can be useful when, for example, you want to use AWS Data Pipeline to process data that is stored inside your organization's firewall. By installing Task Runner on a server in the local network, you can access the local database securely and then poll AWS Data Pipeline for the next task to run. When AWS Data Pipeline ends processing or deletes the pipeline, the Task Runner instance remains running on your computational resource until you manually shut it down. The Task Runner logs persist after pipeline execution is complete.

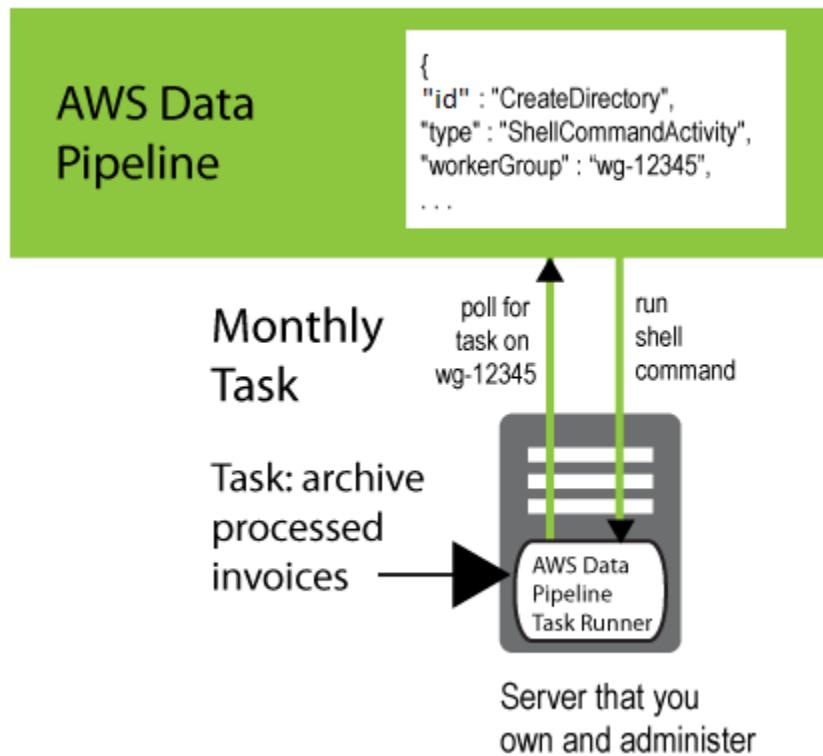
To use Task Runner on a resource that you manage, you must first download Task Runner, and then install it on your computational resource, using the procedures in this section.

Note

You can only install Task Runner on Linux, UNIX, or macOS. Task Runner is not supported on the Windows operating system.

To use Task Runner 2.0, the minimum Java version needed is 1.7.

To connect a Task Runner that you've installed to the pipeline activities it should process, add a `workerGroup` field to the object, and configure Task Runner to poll for that worker group value. You do this by passing the worker group string as a parameter (for example, `--workerGroup=wg-12345`) when you run the Task Runner JAR file.



```
{  
  "id" : ".CreateDirectory",  
  "type" : "ShellCommandActivity",  
  "workerGroup" : "wg-12345",  
  "command" : "mkdir new-directory"  
}
```

Installing Task Runner

This section explains how to install and configure Task Runner and its prerequisites. Installation is a straightforward manual process.

To install Task Runner

1. Task Runner requires Java versions 1.6 or 1.8. 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 1.8 installed on your computer, download one of these versions from <http://www.oracle.com/technetwork/java/index.html>. Download and install Java, and then proceed to the next step.

2. Download TaskRunner-1.0.jar from <https://s3.amazonaws.com/datapipeline-us-east-1/us-east-1/software/latest/TaskRunner/TaskRunner-1.0.jar> and then copy it into a folder on the target computing resource. For Amazon EMR clusters running EmrActivity tasks, install Task Runner on the master node of the cluster.
3. When using Task Runner to connect to the AWS Data Pipeline web service to process your commands, users need programmatic access to a role that has permissions to create or manage data pipelines. For more information, see [Granting programmatic access](#).
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, choose **Instances**, and then select your DB instance.
3. Under **Security and Network**, select the security group, which opens the **Security Groups** page with this DB security group selected. Select 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, choose **Instances**.
3. Select the details icon for the DB instance. Under **Security and Network**, open 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 selecting the icon that's displayed at the top of the console page.
4. On the **Inbound** tab, choose **Edit, 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://amzn-s3-demo-bucket/foldername
```

The **--config** option points to your credentials file.

The **--workerGroup** option specifies the name of your worker group, which must be the same value as specified in your pipeline for tasks to be processed.

The **--region** option specifies the service region from which to pull tasks to execute.

The **--logUri** option is used for pushing your compressed logs to a location in Amazon S3.

When Task Runner is active, it prints the path to where log files are written in the terminal window. The following is an example.

```
Logging to /Computer_Name/.../output/logs
```

Task Runner should be run detached from your login shell. If you are using a terminal application to connect to your computer, you may need to use a utility like nohup or screen to prevent the Task Runner application from exiting when you log out. For more information about command line options, see [Task Runner Configuration Options](#).

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 are two threads allocated from the tasks pool and each thread polls 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 helps to improve the performance of your application. Similarly, applications with long-running preconditions may need to have the timeout and retry values increased. For more information about precondition objects, see [Preconditions](#).

Task Runner Configuration Options

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

Command Line Parameter	Description
--help	Command line help. Example: Java -jar TaskRunner-1.0.jar --help
--config	The path and file name of your credentials.json file.
--accessId	Your AWS access key ID for Task Runner to use when making requests.

Command Line Parameter	Description
	The <code>--accessID</code> and <code>--secretKey</code> options provide an alternative to using a <code>credentials.json</code> file. If a <code>credentials.json</code> file is also provided, the <code>--accessID</code> and <code>--secretKey</code> options take precedence.
<code>--secretKey</code>	Your AWS secret key for Task Runner to use when making requests. For more information, see <code>--accessID</code> .
<code>--endpoint</code>	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 .
<code>--workerGroup</code>	The name of the worker group for which Task Runner retrieves work. Required. When Task Runner polls the web service, it uses the credentials you supplied and the value of <code>workerGroup</code> 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 always get tasks from the region specified in <code>--region</code> .

Command Line Parameter	Description
--taskrunnerId	The ID of the task runner to use when reporting progress. Optional.
--output	The Task Runner directory for log output files. Optional. Log files are stored in a local directory until they are pushed to Amazon S3. This option overrides the default directory.
--region	<p>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, <code>us-east-1</code>.</p> <p>Other supported regions are: <code>eu-west-1</code> , <code>ap-northeast-1</code> , <code>ap-southeast-2</code> , <code>us-west-2</code> .</p>
--logUri	The Amazon S3 destination path for Task Runner to back up log files to every hour. When Task Runner terminates, active logs in the local directory are pushed to the Amazon S3 destination folder.
--proxyHost	The host of the proxy used by Task Runner clients to connect to AWS services.
--proxyPort	Port of the proxy host used by Task Runner clients to connect to AWS services.
--proxyUsername	The user name for proxy.
--proxyPassword	The password for proxy.
--proxyDomain	The Windows domain name for NTLM Proxy.

Command Line Parameter	Description
--proxyWorkstation	The Windows workstation name for NTLM Proxy.

Using Task Runner with a Proxy

If you are using a proxy host, you can either specify its [configuration](#) when invoking Task Runner or set the environment variable, `HTTPS_PROXY`. The environment variable used with Task Runner accepts the same configuration used for the [AWS Command Line Interface](#).

Task Runner and Custom AMIs

When you specify an `Ec2Resource` object for your pipeline, AWS Data Pipeline creates an EC2 instance for you, using an AMI that installs and configures Task Runner for you. A PV-compatible instance type is required in this case. Alternatively, you can create a custom AMI with Task Runner, and then specify the ID of this AMI using the `imageId` field of the `Ec2Resource` object. For more information, see [Ec2Resource](#).

A custom AMI must meet the following requirements for AWS Data Pipeline to use it successfully for Task Runner:

- Create the AMI in the same region in which the instances will run. For more information, see [Creating Your Own AMI](#) in the *Amazon EC2 User Guide*.
- Ensure that the virtualization type of the AMI is supported by the instance type you plan to use. For example, the I2 and G2 instance types require an HVM AMI and the T1, C1, M1, and M2 instance types require a PV AMI. For more information, see [Linux AMI Virtualization Types](#) in the *Amazon EC2 User Guide*.
- Install the following software:
 - Linux
 - Bash
 - wget
 - unzip
 - Java 1.6 or 1.8
 - cloud-init

- Create and configure a user named ec2-user.

Troubleshooting

When you have a problem with AWS Data Pipeline, the most common symptom is that a pipeline doesn't run. You can use the data that the console and CLI provide to identify the problem and find a solution.

Contents

- [Locating Errors in Pipelines](#)
- [Identifying the Amazon EMR Cluster that Serves Your Pipeline](#)
- [Interpreting Pipeline Status Details](#)
- [Locating Error Logs](#)
- [Resolving Common Problems](#)

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, locate the instance pipeline and select the triangle to the left of it, to expand the details.
3. At the bottom of this panel, choose **View execution details**; the **Instance summary** panel opens to show the details of the selected instance.
4. In the **Instance summary** panel, select the triangle next to the instance to see additional details of the instance, and choose **Details, More...**. If the status of your selected instance is **FAILED**, the details box has entries for the error message, the `errorStackTrace` and other information. You can save this information into a file. Choose **OK**.
5. In the **Instance summary** pane, choose **Attempts**, to see details for each attempt row.
6. To take an action on your incomplete or failed instance, select the check box next the instance. This activates the actions. Then, select an action (**Rerun** | **Cancel** | **Mark Finished**).

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, select the triangle next to the pipeline instance, to expand the instance details.
2. Choose **View execution details** and select the triangle next to the component.
3. In the **Details** column, choose **More....** The information screen opens listing the details of the component. Locate and copy the **instanceParent** value from the screen, such as:
@EmrActivityId_xiFDD_2017-09-30T21:40:13
4. Navigate to the Amazon EMR console, search for a cluster with the matching **instanceParent** value in its name, and then choose **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.

5. Now that you know which Amazon EMR cluster contains the error that causes your pipeline failure, follow the [Troubleshooting Tips](#) in the *Amazon EMR Developer Guide*.

Interpreting Pipeline Status Details

The various status levels displayed in the AWS Data Pipeline console and CLI indicate the condition of a pipeline and its components. The pipeline status is simply an overview of a pipeline; to see more information, view the status of individual pipeline components. You can do this by clicking through a pipeline in the console or retrieving pipeline component details using the CLI.

Status Codes

ACTIVATING

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

CANCELED

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

CASCADE_FAILED

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

DEACTIVATING

The pipeline is being deactivated.

FAILED

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

FINISHED

The component completed its assigned work.

INACTIVE

The pipeline was deactivated.

PAUSED

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

PENDING

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

RUNNING

The resource is running and ready to receive work.

SCHEDULED

The resource is scheduled to run.

SHUTTING_DOWN

The resource is shutting down after successfully completing its work.

SKIPPED

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

TIMEDOUT

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

VALIDATING

The pipeline definition is being validated by AWS Data Pipeline.

WAITING_FOR_RUNNER

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

WAITING_ON_DEPENDENCIES

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

Locating Error Logs

This section explains how to find the various logs that AWS Data Pipeline writes, which you can use to determine the source of certain failures and errors.

Pipeline Logs

We recommend that you configure pipelines to create log files in a persistent location, such as in the following example where you use the `pipelineLogUri` field on a pipeline's `Default` object

to cause all pipeline components to use an Amazon S3 log location by default (you can override this by configuring a log location in a specific pipeline component).

Note

Task Runner stores its logs in a different location by default, which may be unavailable when the pipeline finishes and the instance that runs Task Runner terminates. For more information, see [Verifying Task Runner Logging](#).

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

```
{ "objects": [  
  {  
    "id":"Default",  
    "pipelineLogUri":"s3://amzn-s3-demo-bucket/error_logs"  
  },  
  ...  
]
```

After you configure a pipeline log directory, Task Runner creates a copy of the logs in your directory, with the same formatting and file names described in the previous section about Task Runner logs.

Hadoop Job and Amazon EMR Step Logs

With any Hadoop-based activity such as [HadoopActivity](#), [HiveActivity](#), or [PigActivity](#) you can view Hadoop job logs at the location returned in the runtime slot, `hadoopJobLog`. [EmrActivity](#) has its own logging features and those logs are stored using the location chosen by Amazon EMR and returned by the runtime slot, `emrStepLog`. For more information, see [View Log Files](#) in the Amazon EMR Developer Guide.

Resolving Common Problems

This topic provides various symptoms of AWS Data Pipeline problems and the recommended steps to solve them.

Contents

- [Pipeline Stuck in Pending Status](#)

- [Pipeline Component Stuck in Waiting for Runner Status](#)
- [Pipeline Component Stuck in WAITING_ON_DEPENDENCIES Status](#)
- [Run Doesn't Start When Scheduled](#)
- [Pipeline Components Run in Wrong Order](#)
- [EMR Cluster Fails With Error: The security token included in the request is invalid](#)
- [Insufficient Permissions to Access Resources](#)
- [Status Code: 400 Error Code: PipelineNotFoundException](#)
- [Creating a Pipeline Causes a Security Token Error](#)
- [Cannot See Pipeline Details in the Console](#)
- [Error in remote runner Status Code: 404, AWS Service: Amazon S3](#)
- [Access Denied - Not Authorized to Perform Function datapipeline:](#)
- [Older Amazon EMR AMIs May Create False Data for Large CSV Files](#)
- [Increasing AWS Data Pipeline Limits](#)

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:

```
aws datapipeline --get-pipeline-definition --pipeline-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 does the 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 can move out of the WAITING_ON_DEPENDENCIES state.

For example, consider the following excerpt from a pipeline definition. In this case, the InputData object has a precondition 'Ready' specifying that the data must exist before the InputData object is complete. If the data does not exist, the InputData object remains in the WAITING_ON_DEPENDENCIES state, waiting for the data specified by the path field to become available. Any objects that depend on InputData likewise remain in a WAITING_ON_DEPENDENCIES state waiting for the InputData object to reach the FINISHED state.

```
{  
    "id": "InputData",  
    "type": "S3DataNode",  
    "filePath": "s3://elasticmapreduce/samples/wordcount/wordSplitter.py",  
    "schedule": {"ref": "MySchedule"},  
    "precondition": "Ready"  
},
```

```
{  
  "id": "Ready",  
  "type": "Exists"  
  ...
```

Also, check that your objects have the proper permissions to access the data. In the preceding example, if the information in the credentials field did not have permissions to access the data specified in the path field, the InputData object would get stuck in a WAITING_ON_DEPENDENCIES state because it cannot access the data specified by the path field, even if that data exists.

It is also possible that a resource communicating with Amazon S3 does not have a public IP address associated with it. For example, an Ec2Resource in a public subnet must have a public IP address associated with it.

Lastly, under certain conditions, resource instances can reach the WAITING_ON_DEPENDENCIES state much earlier than their associated activities are scheduled to start, which may give the impression that the resource or the activity is failing.

Run Doesn't Start When Scheduled

Check that you chose the correct schedule type that determines whether your task starts at the beginning of the schedule interval (Cron Style Schedule Type) or at the end of the schedule interval (Time Series Schedule Type).

Additionally, check that you have properly specified the dates in your schedule objects and that the startDateTime and endDateTime values are in UTC format, such as in the following example:

```
{  
  "id": "MySchedule",  
  "startDateTime": "2012-11-12T19:30:00",  
  "endDateTime": "2012-11-12T20:30:00",  
  "period": "1 Hour",  
  "type": "Schedule"  
},
```

Pipeline Components Run in Wrong Order

You might notice that the start and end times for your pipeline components are running in the wrong order, or in a different sequence than you expect. It is important to understand that pipeline components can start running simultaneously if their preconditions are met at start-up time. In

other words, pipeline components do not execute sequentially by default; if you need a specific execution order, you must control the execution order with `preconditions` and `dependsOn` fields.

Verify that you are using the `dependsOn` field populated with a reference to the correct prerequisite pipeline components, and that all the necessary pointers between components are present to achieve the order you require.

EMR Cluster Fails With Error: The security token included in the request is invalid

Verify your IAM roles, policies, and trust relationships as described in [IAM Roles for AWS Data Pipeline](#).

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 the creation of resources on your behalf. For example, when you create a pipeline that uses an EC2 instance to run a command to move data, AWS Data Pipeline can provision this EC2 instance for you. If you encounter problems, especially those involving resources that you can access manually but AWS Data Pipeline cannot, verify your IAM roles, policies, and trust relationships as described in [IAM Roles for AWS Data Pipeline](#).

Status Code: 400 Error Code: PipelineNotFoundException

This error means that your IAM default roles might not have the required permissions necessary for AWS Data Pipeline to function correctly. For more information, see [IAM Roles for AWS Data Pipeline](#).

Creating a Pipeline Causes a Security Token Error

You receive the following error when you try to create a pipeline:

Failed to create pipeline with 'pipeline_name'. Error: UnrecognizedClientException - The security token included in the request is invalid.

Cannot See Pipeline Details in the Console

The AWS Data Pipeline console pipeline filter applies to the *scheduled start date* for a pipeline, without regard to when the pipeline was submitted. It is possible to submit a new pipeline using a

scheduled start date that occurs in the past, which the default date filter may not show. To see the pipeline details, change your date filter to ensure that the scheduled pipeline start date fits within the date range filter.

Error in remote runner Status Code: 404, AWS Service: Amazon S3

This error means that Task Runner could not access your files in Amazon S3. Verify that:

- You have credentials correctly set
- The Amazon S3 bucket that you are trying to access exists
- You are authorized to access the Amazon S3 bucket

Access Denied - Not Authorized to Perform Function datapipeline:

In the Task Runner logs, you may see an error that is similar to the following:

- ERROR Status Code: 403
- AWS Service: DataPipeline
- AWS Error Code: AccessDenied
- AWS Error Message: User: arn:aws:sts::XXXXXXXXXXXX:federated-user/i-XXXXXXX is not authorized to perform: datapipeline:PollForTask.

 **Note**

In this error message, PollForTask may be replaced with names of other AWS Data Pipeline permissions.

This error message indicates that the IAM role you specified needs additional permissions necessary to interact with AWS Data Pipeline. Ensure that your IAM role policy contains the following lines, where PollForTask is replaced with the name of the permission you want to add (use * to grant all permissions). For more information about how to create a new IAM role and apply a policy to it, see [Managing IAM Policies](#) in the *Using IAM* guide.

```
{  
  "Action": [ "datapipeline:PollForTask" ],  
  "Effect": "Allow",
```

```
"Resource": ["*"]  
}
```

Older Amazon EMR AMIs May Create False Data for Large CSV Files

On Amazon EMR AMIs previous to 3.9 (3.8 and below) AWS Data Pipeline uses a custom InputFormat to read and write CSV files for use with MapReduce jobs. This is used when the service stages tables to and from Amazon S3. An issue with this InputFormat was discovered where reading records from large CSV files may result in producing tables that are not correctly copied. This issue was fixed in later Amazon EMR releases. Please use Amazon EMR AMI 3.9 or an Amazon EMR release 4.0.0 or greater.

Increasing AWS Data Pipeline Limits

Occasionally, you may exceed specific AWS Data Pipeline system limits. For example, the default pipeline limit is 20 pipelines with 50 objects in each. If you discover that you need more pipelines than the limit, consider merging multiple pipelines to create fewer pipelines with more objects in each. For more information about the AWS Data Pipeline limits, see [AWS Data Pipeline Limits](#). 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](#).

AWS Data Pipeline Limits

To ensure that there is capacity for all users, AWS Data Pipeline imposes limits on the resources that you can allocate and the rate at which you can allocate resources.

Contents

- [Account Limits](#)
- [Web Service Call Limits](#)
- [Scaling Considerations](#)

Account Limits

The following limits apply to a single AWS account. If you require additional capacity, you can use the [Amazon Web Services Support Center request form](#) to increase your capacity.

Attribute	Limit	Adjustable
Number of pipelines	100	Yes
Number of objects per pipeline	100	Yes
Number of active instances per object	5	Yes
Number of fields per object	50	No
Number of UTF8 bytes per field name or identifier	256	No
Number of UTF8 bytes per field	10,240	No
Number of UTF8 bytes per object	15,360 (including field names)	No

Attribute	Limit	Adjustable
Rate of creation of an instance from an object	1 per 5 minutes	No
Retries of a pipeline activity	5 per task	No
Minimum delay between retry attempts	2 minutes	No
Minimum scheduling interval	15 minutes	No
Maximum number of roll-ups into a single object	32	No
Maximum number of EC2 instances per Ec2Resource object	1	No

Web Service Call Limits

AWS Data Pipeline limits the rate at which you can call the web service API. These limits also apply to AWS Data Pipeline agents that call the web service API on your behalf, such as the console, CLI, and Task Runner.

The following limits apply to a single AWS account. This means the total usage on the account, including that by 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 one call each five seconds. If you don't call the service for 30 seconds, you have six calls saved up. You could then call the web service six times in a second. Because this is below the burst limit and keeps your average calls at the regular rate limit, your calls are not throttled.

If you exceed the rate limit and the burst limit, your web service call fails and returns a throttling exception. The default implementation of a worker, Task Runner, automatically retries API calls that fail with a throttling exception. Task Runner has a back off so that subsequent attempts to call the API occur at increasingly longer intervals. If you write a worker, we recommend that you implement similar retry logic.

These limits are applied against an individual AWS account.

API	Regular rate limit	Burst limit
ActivatePipeline	1 call per second	100 calls
CreatePipeline	1 call per second	100 calls
DeletePipeline	1 call per second	100 calls
DescribeObjects	2 calls per second	100 calls
DescribePipelines	1 call per second	100 calls
GetPipelineDefinition	1 call per second	100 calls
PollForTask	2 calls per second	100 calls
ListPipelines	1 call per second	100 calls
PutPipelineDefinition	1 call per second	100 calls
QueryObjects	2 calls per second	100 calls
ReportTaskProgress	10 calls per second	100 calls
SetTaskStatus	10 calls per second	100 calls
setStatus	1 call per second	100 calls
ReportTaskRunnerHeartbeat	1 call per second	100 calls
ValidatePipelineDefinition	1 call per second	100 calls

Scaling Considerations

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

If you require additional capacity, you can use the [Amazon Web Services Support Center request form](#) to increase your capacity.

AWS Data Pipeline Resources

The following are resources to help you use AWS Data Pipeline.

- [**AWS Data Pipeline Product Information**](#)—The primary web page for information about AWS Data Pipeline.
- [**AWS Data Pipeline Technical FAQ**](#) – Covers the top 20 questions developers ask about this product.
- [**Release Notes**](#) – Provide a high-level overview of the current release. They specifically note any new features, corrections, and known issues.
- [**AWS Data Pipeline Discussion Forums**](#) – A community-based forum for developers to discuss technical questions related to Amazon Web Services.
- [**Classes & Workshops**](#) – Links to role-based and specialty courses, in addition to self-paced labs to help sharpen your AWS skills and gain practical experience.
- [**AWS Developer Center**](#) – Explore tutorials, download tools, and learn about AWS developer events.
- [**AWS Developer Tools**](#) – Links to developer tools, SDKs, IDE toolkits, and command line tools for developing and managing AWS applications.
- [**Getting Started Resource Center**](#) – Learn how to set up your AWS account, join the AWS community, and launch your first application.
- [**Hands-On Tutorials**](#) – Follow step-by-step tutorials to launch your first application on AWS.
- [**AWS Whitepapers**](#) – Links to a comprehensive list of technical AWS whitepapers, covering topics such as architecture, security, and economics and authored by AWS Solutions Architects or other technical experts.
- [**AWS Support Center**](#) – The hub for creating and managing your AWS Support cases. Also includes links to other helpful resources, such as forums, technical FAQs, service health status, and AWS Trusted Advisor.
- [**Support**](#) – The primary webpage for information about 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.

Change	Description	Release Date
AWS Data Pipeline is no longer available to new customers	AWS Data Pipeline is no longer available to new customers. Existing customers of AWS Data Pipeline can continue to use the service as normal. Learn more	25 July 2025
Added documentation for performing certain procedures using the AWS CLI. Removed AWS Data Pipeline console related procedures.	For more information, see Cloning Your Pipeline , Viewing Pipeline Logs , and Create a pipeline from Data Pipeline templates using the CLI .	26 May 2023
Added more content and samples for migrating from AWS Data Pipeline to other alternative services.	Updated the topic for migrating AWS Data Pipeline to either AWS Glue, AWS Step Functions, or Amazon MWAA with more information on each alternative, concept mappings between the services, and samples. For more information, see Migrating workloads from AWS Data Pipeline .	31 March 2023
Added information on AWS Data Pipeline support of IMDSv2.	AWS Data Pipeline supports IMDSv2 for Amazon EMR and Amazon EC2 resources. For more information, see Data Protection in AWS Data Pipeline , EmrCluster , and Ec2Resource .	16 December 2022
Added a topic for migrating from AWS Data Pipeline to other alternative services.	There are now other AWS services that offer customers a better data integration experience. You can migrate typical use cases of AWS Data Pipeline to either AWS Glue, AWS Step Functions, or Amazon MWAA. For more information, see Migrating workloads from AWS Data Pipeline .	16 December 2022

Change	Description	Release Date
<p>Updated the lists of supported Amazon EC2 and Amazon EMR instances.</p> <p>Updated the list of IDs of the HVM (Hardware Virtual Machine) AMIs used for the instances.</p>	<p>Updated the lists of supported Amazon EC2 and Amazon EMR instances. For more information, see Supported Instance Types for Pipeline Work Activities.</p> <p>Updated the list of IDs of the HVM (Hardware Virtual Machine) AMIs used for the instances. For more information, see Syntax and search for <code>imageId</code>.</p>	9 November 2018
<p>Added configuration for attaching Amazon EBS volumes to cluster nodes, and for launching an Amazon EMR cluster into a private subnet.</p>	<p>Added configuration options to an <code>EMRcluster</code> object. You can use these options in pipelines that use Amazon EMR clusters.</p> <p>Use the <code>coreEbsConfiguration</code>, <code>masterEbsConfiguration</code>, and <code>TaskEbsConfiguration</code> fields to configure the attachment of Amazon EBS volumes to core, master, and task nodes in the Amazon EMR cluster. For more information, see Attach EBS volumes to cluster nodes.</p> <p>Use the <code>emrManagedMasterSecurityGroupId</code>, <code>emrManagedSlaveSecurityGroupId</code>, and <code>ServiceAccessSecurityGroupId</code> fields to configure an Amazon EMR cluster in a private subnet. For more information, see Configure an Amazon EMR cluster in a private subnet.</p> <p>For more information about <code>EMRcluster</code> syntax, see EmrCluster.</p>	19 April 2018

Change	Description	Release Date
Added the list of supported Amazon EC2 and Amazon EMR instances.	<p>Added the list of instances that AWS Data Pipeline creates by default, if you do not specify an instance type in the pipeline definition. Added a list of supported Amazon EC2 and Amazon EMR instances.</p> <p>For more information, see Supported Instance Types for Pipeline Work Activities.</p>	22 March 2018
Added support for On-demand pipelines.	<ul style="list-style-type: none"> Added support for On-demand pipelines, which allows you to re-run a pipeline by activating it again. 	22 February 2016
Additional support for RDS databases	<ul style="list-style-type: none"> Added <code>rdsInstanceId</code>, <code>region</code>, and <code>jdbcDriverJarUri</code> to RdsDatabase. Updated database in SqlActivity to also support RdsDatabase. 	17 August 2015
Additional JDBC support	<ul style="list-style-type: none"> Updated database in SqlActivity to also support JdbcDatabase. Added <code>jdbcDriverJarUri</code> to JdbcDatabase. Added <code>initTimeout</code> to Ec2Resource and EmrCluster. Added <code>runAsUser</code> to Ec2Resource. 	7 July 2015
HadoopActivity, Availability Zone, and Spot Support	<ul style="list-style-type: none"> Added support for submitting parallel work to Hadoop clusters. For more information, see HadoopActivity. Added the ability to request Spot Instances with Ec2Resource and EmrCluster. Added the ability to launch EmrCluster resources in a specified Availability Zone. 	1 June 2015

Change	Description	Release Date
Deactivating pipelines	Added support for deactivating active pipelines. For more information, see Deactivating Your Pipeline .	7 April 2015
Updated templates and console	Added new templates. Updated the Getting Started chapter to use the Getting Started with ShellComm andActivity template. For more information, see Create a pipeline from Data Pipeline templates using the CLI .	25 November 2014
VPC support	Added support for launching resources into a virtual private cloud (VPC).	12 March 2014
Region support	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> .	20 February 2014
Amazon Redshift support	Added support for Amazon 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 , RedshiftDataNode , RedshiftDatabase , and RedshiftCopyActivity .	6 November 2013
PigActivity	Added PigActivity, which provides native support for Pig. For more information, see PigActivity .	15 October 2013
New console template, activity, and data format	Added the new CrossRegion DynamoDB Copy console template, including the new HiveCopyActivity and DynamoDBExportDataFormat.	21 August 2013
Cascading failures and reruns	Added information about AWS Data Pipeline cascading failure and rerun behavior. For more information, see Cascading failures and reruns .	8 August 2013

Change	Description	Release Date
Troubleshooting video	Added the AWS Data Pipeline Basic Troubleshooting video. For more information, see Troubleshooting .	17 July 2013
Editing active pipelines	Added more information about editing active pipelines and rerunning pipeline components. For more information, see Editing Your Pipeline .	17 July 2013
Use resources in different regions	Added more information about using resources in different regions. For more information, see Using a Pipeline with Resources in Multiple Regions .	17 June 2013
WAITING_O_N_DEPENDENCIES status	CHECKING_PRECONDITIONS status changed to WAITING_ON_DEPENDENCIES and added the @waitingOn runtime field for pipeline objects.	20 May 2013
DynamoDBDataFormat	Added DynamoDBDataFormat template.	23 April 2013
Process Web Logs video and Spot Instances support	Introduced the video "Process Web Logs with AWS Data Pipeline, Amazon EMR, and Hive," and Amazon EC2 Spot Instances support.	21 February 2013
	The initial release of the AWS Data Pipeline Developer Guide.	20 December 2012