
Amazon EMR

Management Guide



Amazon EMR: Management Guide

Copyright © 2017 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

What is Amazon EMR?	1
Overview	1
Understanding Clusters and Nodes	2
Submitting Work to a Cluster	2
Processing Data	3
Understanding the Cluster Lifecycle	4
Benefits	5
Cost Savings	6
AWS Integration	6
Deployment	6
Scalability and Flexibility	7
Reliability	7
Security	7
Monitoring	8
Management Interfaces	9
Architecture	9
Storage	9
Cluster Resource Management	10
Data Processing Frameworks	10
Applications and Programs	11
Getting Started	12
Step 1: Set Up Prerequisites	13
Sign Up for AWS	13
Create an Amazon S3 Bucket	13
Create an Amazon EC2 Key Pair	13
Step 2: Launch Your Sample Cluster	14
Using Quick Cluster Configuration Overview	14
Launch the Sample Cluster	20
Step 3: Prepare Your Sample Data and Script	20
Sample Data Overview	20
Sample Hive Script Overview	21
Step 4: Process Your Sample Data	22
Submit the Hive Script as a Step	22
View the Results	23
Step 5: Reset Your Environment	23
Plan and Configure Clusters	24
Configure Cluster Location and Data Storage	24
Choose an AWS Region	25
Work with Storage and File Systems	26
Prepare Input Data	44
Configure an Output Location	52
Configure a Cluster to be Transient or Long-Running	56
Using a Custom AMI	57
Best Practices and Considerations	58
Specifying a Custom AMI	58
Managing AMI Package Repository Updates	59
Creating a Custom Amazon Linux AMI from a Preconfigured Instance	59
Creating a Custom AMI with an Encrypted Amazon EBS Root Device Volume	61
Specifying the Amazon EBS Root Device Volume Size	62
Configure Cluster Software	63
(Optional) Create Bootstrap Actions to Install Additional Software	64
Configure Cluster Hardware and Networking	67
Master Node	67
Core Nodes	67

Task Nodes	68
Instance Fleets	68
Uniform Instance Groups	68
Plan and Configure EC2 Instances	68
Plan and Configure Networking	73
Create a Cluster with Instance Fleets or Uniform Instance Groups	85
Configure Access to the Cluster	99
Configure User Permissions Using IAM Roles	99
Configure IAM Roles for Amazon EMR and Applications	106
Configure Security Groups	113
Create SSH Credentials for the Master Node	119
Setting Permissions on the System Directory	120
Configure Cluster Logging and Debugging	120
Default Log Files	121
Archive Log Files to Amazon S3	121
Enable the Debugging Tool	123
Debugging Option Information	124
Tag Clusters	124
Tag Restrictions	125
Tag Resources for Billing	126
Add Tags to a New Cluster	126
Adding Tags to an Existing Cluster	127
View Tags on a Cluster	127
Remove Tags from a Cluster	128
Drivers and Third-Party Application Integration	129
Use Business Intelligence Tools with Amazon EMR	129
Using the MapR Distribution for Hadoop	129
Manage Clusters	139
View and Monitor a Cluster	139
View Cluster Details	140
Enhanced Step Debugging	144
View Log Files	146
View Cluster Instances in Amazon EC2	150
CloudWatch Events and Metrics	151
View Cluster Application Metrics with Ganglia	171
Logging Amazon EMR API Calls in AWS CloudTrail	171
Connect to the Cluster	172
Connect to the Master Node Using SSH	173
View Web Interfaces Hosted on Amazon EMR Clusters	177
Control Cluster Termination	185
Terminate a Cluster	186
Managing Cluster Termination	188
Scaling Cluster Resources	190
Using Automatic Scaling in Amazon EMR	191
Manually Resizing a Running Cluster	199
Configure Cluster Scale-Down	204
Cloning a Cluster Using the Console	205
Submit Work to a Cluster	206
Work with Steps Using the CLI and Console	206
Submit Hadoop Jobs Interactively	208
Add More than 256 Steps to a Cluster	210
Automate Recurring Clusters with AWS Data Pipeline	210
Troubleshoot a Cluster	211
What Tools are Available for Troubleshooting?	211
Tools to Display Cluster Details	211
Tools to View Log Files	212
Tools to Monitor Cluster Performance	212

Troubleshoot a Failed Cluster	212
Step 1: Gather Data About the Issue	213
Step 2: Check the Environment	213
Step 3: Look at the Last State Change	214
Step 4: Examine the Log Files	214
Step 5: Test the Cluster Step by Step	215
Troubleshoot a Slow Cluster	216
Step 1: Gather Data About the Issue	216
Step 2: Check the Environment	217
Step 3: Examine the Log Files	218
Step 4: Check Cluster and Instance Health	219
Step 5: Check for Arrested Groups	220
Step 6: Review Configuration Settings	220
Step 7: Examine Input Data	222
Common Errors in Amazon EMR	222
Input and Output Errors	222
Permissions Errors	224
Resource Errors	225
Streaming Cluster Errors	228
Custom JAR Cluster Errors	229
Hive Cluster Errors	230
VPC Errors	231
AWS GovCloud (US) Errors	233
Other Issues	234
Write Applications that Launch and Manage Clusters	235
End-to-End Amazon EMR Java Source Code Sample	235
Common Concepts for API Calls	238
Endpoints for Amazon EMR	238
Specifying Cluster Parameters in Amazon EMR	238
Availability Zones in Amazon EMR	239
How to Use Additional Files and Libraries in Amazon EMR Clusters	239
Amazon EMR Sample Applications	239
Use SDKs to Call Amazon EMR APIs	239
Using the AWS SDK for Java to Create an Amazon EMR Cluster	240
Using the Java SDK to Sign an API Request	241

What is Amazon EMR?

This documentation is for versions 4.x and 5.x of Amazon EMR. For information about Amazon EMR AMI versions 2.x and 3.x, see the [Amazon EMR Developer Guide \(PDF\)](#).

Amazon EMR is a managed cluster platform that simplifies running big data frameworks, such as [Apache Hadoop](#) and [Apache Spark](#), on AWS to process and analyze vast amounts of data. By using these frameworks and related open-source projects, such as Apache Hive and Apache Pig, you can process data for analytics purposes and business intelligence workloads. Additionally, you can use Amazon EMR to transform and move large amounts of data into and out of other AWS data stores and databases, such as Amazon Simple Storage Service (Amazon S3) and Amazon DynamoDB.

If you are a first-time user of Amazon EMR, we recommend that you begin by reading the following:

- [What is Amazon EMR? \(p. 1\)](#) (*this section*) – This section provides an overview of Amazon EMR functionality and features.
- [Amazon EMR](#) – This service page provides the Amazon EMR highlights, product details, and pricing information.
- [Getting Started: Analyzing Big Data with Amazon EMR \(p. 12\)](#) – This section provides a tutorial of using Amazon EMR to create a sample cluster and run a Hive script as a step.

In This Section

- [Overview of Amazon EMR \(p. 1\)](#)
- [Benefits of Using Amazon EMR \(p. 5\)](#)
- [Overview of Amazon EMR Architecture \(p. 9\)](#)

Overview of Amazon EMR

This topic provides an overview of Amazon EMR clusters, including how to submit work to a cluster, how that data is processed, and the various states that the cluster goes through during processing.

In This Topic

- [Understanding Clusters and Nodes \(p. 2\)](#)
- [Submitting Work to a Cluster \(p. 2\)](#)
- [Processing Data \(p. 3\)](#)
- [Understanding the Cluster Lifecycle \(p. 4\)](#)

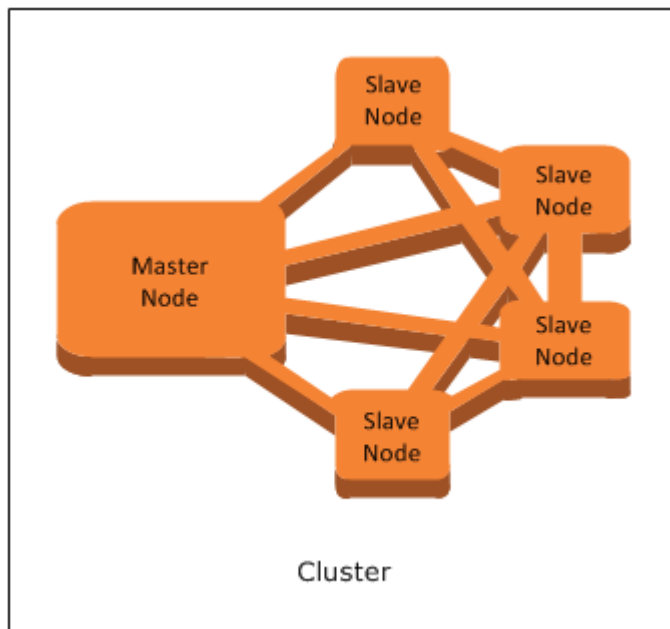
Understanding Clusters and Nodes

The central component of Amazon EMR is the *cluster*. A cluster is a collection of Amazon Elastic Compute Cloud (Amazon EC2) instances. Each instance in the cluster is called a *node*. Each node has a role within the cluster, referred to as the *node type*. Amazon EMR also installs different software components on each node type, giving each node a role in a distributed application like Apache Hadoop.

The node types in Amazon EMR are as follows:

- **Master node:** a node that manages the cluster by running software components which coordinate the distribution of data and tasks among other nodes—collectively referred to as slave nodes—for processing. The master node tracks the status of tasks and monitors the health of the cluster.
- **Core node:** a slave node that has software components which run tasks and store data in the Hadoop Distributed File System (HDFS) on your cluster.
- **Task node:** a slave node that has software components which only run tasks. Task nodes are optional.

The following diagram represents a cluster with one master node and four slave nodes.



Submitting Work to a Cluster

When you run your cluster on Amazon EMR, you have several options as to how you specify the work that needs to be done.

- Provide the entire definition of the work to be done in the Map and Reduce functions. This is typically done for clusters that process a set amount of data and then terminate when processing is complete. For more information, see [Apache Hadoop](#) in the *Amazon EMR Release Guide*.

- Create a long-running cluster and use the Amazon EMR console, the Amazon EMR API, or the AWS CLI to submit steps, which may contain one or more Hadoop jobs. For more information, see [Submit Work to a Cluster \(p. 206\)](#).
- Create a cluster with a Hadoop application, such as Hive or Pig, installed and use the interface provided by the application to submit queries, either scripted or interactively. For more information, see the [Amazon EMR Release Guide](#).
- Create a long-running cluster, connect to it, and submit Hadoop jobs using the Hadoop API. For more information, go to [Class JobClient](#) in the Apache Hadoop API documentation.

Processing Data

When you launch your cluster, you choose the frameworks and applications to install for your data processing needs. There are two ways to process data in your Amazon EMR cluster: by submitting jobs or queries directly to the applications that are installed on your cluster or by running *steps* in the cluster.

Submitting Jobs Directly to Applications

You can submit jobs and interact directly with the software that is installed in your Amazon EMR cluster. To do this, you typically connect to the master node over a secure connection and access the interfaces and tools that are available for the software that runs directly on your cluster. For more information, see [Connect to the Cluster \(p. 172\)](#).

Running Steps to Process Data

You can submit one or more ordered steps to an Amazon EMR cluster. Each step is a unit of work that contains instructions to manipulate data for processing by software installed on the cluster.

The following is an example process using four steps:

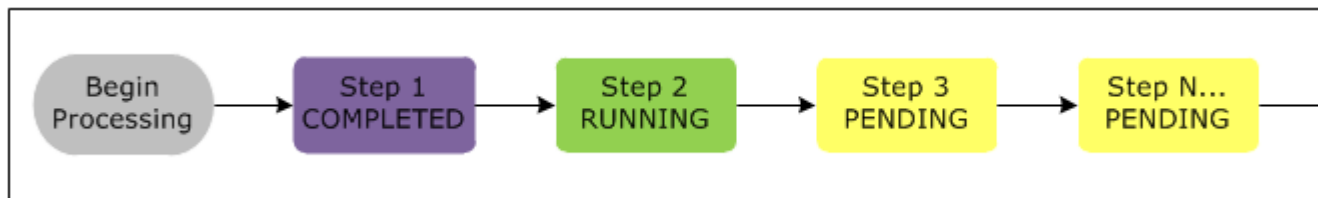
1. Submit an input data set for processing
2. Process the output of the first step by using a Pig program
3. Process a second input data set by using a Hive program
4. Write an output data set

Generally, when you process data in Amazon EMR, the input is data stored as files in your chosen underlying file system, such as Amazon S3 or HDFS. This data passes from one step to the next in the processing sequence. The final step writes the output data to a specified location, such as an Amazon S3 bucket.

Steps are run in the following sequence:

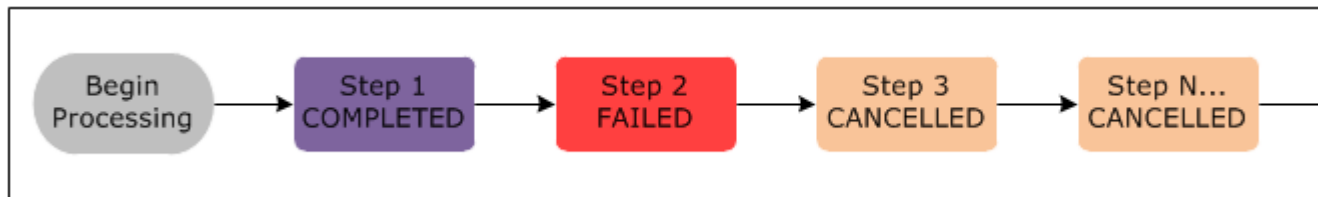
1. A request is submitted to begin processing steps.
2. The state of all steps is set to **PENDING**.
3. When the first step in the sequence starts, its state changes to **RUNNING**. The other steps remain in the **PENDING** state.
4. After the first step completes, its state changes to **COMPLETED**.
5. The next step in the sequence starts, and its state changes to **RUNNING**. When it completes, its state changes to **COMPLETED**.
6. This pattern repeats for each step until they all complete and processing ends.

The following diagram represents the step sequence and change of state for the steps as they are processed.



If a step fails during processing, its state changes to **FAILED**. By default, any remaining steps in the sequence are set to **CANCELLED** and do not run, although you can choose to ignore processing failures and allow remaining steps to proceed.

The following diagram represents the step sequence and default change of state when a step fails during processing.



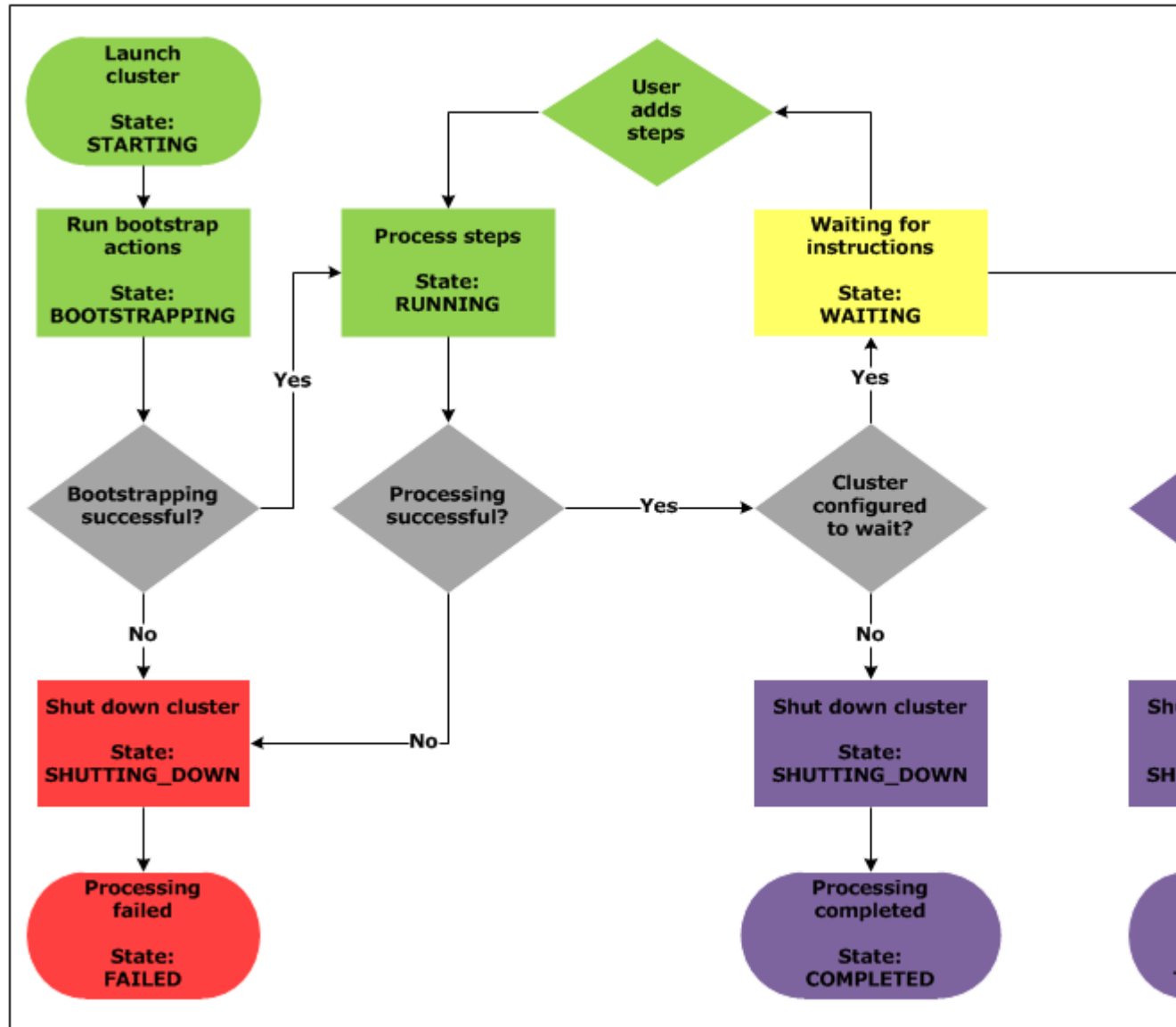
Understanding the Cluster Lifecycle

A successful Amazon EMR cluster follows this process:

1. Amazon EMR first provisions a cluster with your chosen applications, such as Hadoop or Spark. During this phase, the cluster state is `STARTING`.
2. Next, any user-defined actions—called *bootstrap actions*—such as installing additional applications, run on the cluster. During this phase, the cluster state is `BOOTSTRAPPING`.
3. After all bootstrap actions are successfully completed, the cluster state is `RUNNING`. The cluster sequentially runs all steps during this phase.
4. After steps run successfully, the cluster either goes into a `WAITING` state or a `SHUTTING_DOWN` state, described as follows.
 - If you configured your cluster as a long-running cluster by disabling auto-terminate or by using the `KeepJobFlowAliveWhenNoSteps` parameter in the API, the cluster will go into a `WAITING` state after processing all submitted steps and wait for the next set of instructions. If you have more data to process, you can add more steps. You must manually terminate a long-running cluster when you no longer need it. After you manually terminate the cluster, it goes into the `SHUTTING_DOWN` state and then into the `TERMINATED` state.
 - If you configured your cluster as a transient cluster by enabling auto-terminate or by using the `KeepJobFlowAliveWhenNoSteps` parameter in the API, it automatically goes into the `SHUTTING_DOWN` state after all of the steps complete. The instances are terminated, and all data stored in the cluster itself is deleted. Information stored in other locations, such as in your Amazon S3 bucket, persists. When the shutdown process completes, the cluster state is set to `COMPLETED`.

A failure during the cluster process terminates the cluster and all of its instances unless: a) you enable termination protection; or b) you supply an `ActionOnFailure` in the [StepConfig](#) for the step. Any data stored on the cluster is deleted. The cluster state is set to `FAILED`. If you have enabled termination protection so that you can retrieve data from your cluster in the event of a failure, then the cluster is not terminated. Once you are truly done with the cluster, you can remove termination protection and terminate the cluster. For more information, see [Managing Cluster Termination](#) (p. 188).

The following diagram represents the lifecycle of a cluster, and how each stage of the lifecycle maps to a particular cluster state.



For a complete list of cluster states, go to the [JobFlowExecutionStatusDetail data type](#) in the *Amazon EMR API Reference*. For more information about submitting steps and configuring the cluster lifecycle, see [Submitting Work to a Cluster \(p. 2\)](#) and [Configure a Cluster to be Transient or Long-Running \(p. 56\)](#).

Benefits of Using Amazon EMR

There are many benefits to using Amazon EMR. This section provides an overview of these benefits and links to additional information to help you explore further.

Topics

- [Cost Savings \(p. 6\)](#)
- [AWS Integration \(p. 6\)](#)
- [Deployment \(p. 6\)](#)

- [Scalability and Flexibility \(p. 7\)](#)
- [Reliability \(p. 7\)](#)
- [Security \(p. 7\)](#)
- [Monitoring \(p. 8\)](#)
- [Management Interfaces \(p. 9\)](#)

Cost Savings

Amazon EMR pricing depends on the instance type and number of EC2 instances that you deploy and the region in which you launch your cluster. On-demand pricing offers low hourly rates, but you can reduce the cost even further by purchasing Reserved instances or bidding on Spot instances. Spot instances can offer significant savings—as low as a tenth of on-demand pricing in some cases.

Note

If you use Amazon S3, Amazon Kinesis, or DynamoDB with your EMR cluster, there are additional charges for those services that are billed separately from your Amazon EMR usage. Also, if you install Splunk Hunk or use MapR M5 or M7 distributions on your cluster, there are charges in addition to your Amazon EMR usage.

For more information about pricing options and details, go to [Amazon EMR Pricing](#).

AWS Integration

Amazon EMR integrates with other AWS services to provide capabilities and functionality related to networking, storage, security, and so on for your cluster. The following list provides several examples of this integration:

- Amazon EC2 for the instances that comprise the nodes in the cluster
- Amazon Virtual Private Cloud (Amazon VPC) to configure the virtual network in which you launch your instances
- Amazon S3 to store input and output data
- Amazon CloudWatch to monitor cluster performance and configure alarms
- AWS Identity and Access Management (IAM) to configure permissions
- AWS CloudTrail to audit requests made to the service
- AWS Data Pipeline to schedule and start your clusters

Deployment

Your EMR cluster consists of EC2 instances, which perform the work that you submit to your cluster. When you launch your cluster, Amazon EMR configures the instances with the applications that you choose, such as Apache Hadoop or Spark. Choose the instance size and type that best suits the processing needs for your cluster: batch processing, low-latency queries, streaming data, or large data storage. For more information about the instance types available for Amazon EMR, see [Configure Cluster Hardware and Networking \(p. 67\)](#).

Amazon EMR offers a variety of ways to configure software on your cluster. For example, you can install an Amazon EMR release with a chosen set of applications that can include versatile frameworks, such as Hadoop, and applications, such as Hive, Pig, or Spark. You can also install one of several MapR distributions. Amazon EMR uses Amazon Linux, so you can also install software on your cluster manually using the yum package manager or from the source. For more information, see [Configure Cluster Software \(p. 63\)](#).

Scalability and Flexibility

Amazon EMR provides flexibility to scale your cluster up or down as your computing needs change. You can resize your cluster to add instances for peak workloads and remove instances to control costs when peak workloads subside. For more information, see [Manually Resizing a Running Cluster \(p. 199\)](#).

Amazon EMR also provides the option to run multiple instance groups so that you can use On-Demand instances in one group for guaranteed processing power together with Spot instances in another group to have your jobs completed faster and for lower costs. You can also mix different instance types to take advantage of better pricing for one Spot instance type over another. For more information, see [When Should You Use Spot Instances? \(p. 96\)](#).

Additionally, Amazon EMR provides the flexibility to use several file systems for your input, output, and intermediate data. For example, you might choose the Hadoop Distributed File System (HDFS) which runs on the master and core nodes of your cluster for processing data that you do not need to store beyond your cluster's lifecycle. You might choose the EMR File System (EMRFS) to use Amazon S3 as a data layer for applications running on your cluster so that you can separate your compute and storage, and persist data outside of the lifecycle of your cluster. EMRFS provides the added benefit of allowing you to scale up or down for your compute and storage needs independently. You can scale your compute needs by resizing your cluster and you can scale your storage needs by using Amazon S3. For more information, see [Work with Storage and File Systems \(p. 26\)](#).

Reliability

Amazon EMR monitors nodes in your cluster and automatically terminates and replaces an instance if there is a failure.

Amazon EMR provides configuration options that control how your cluster is terminated—automatically or manually. If you configure your cluster to be automatically terminated, it is terminated after all the steps complete. This is referred to as a transient cluster. However, you can configure the cluster to continue running after processing completes so that you can choose to terminate it manually when you no longer need it. Or, you can create a cluster, interact with the installed applications directly, and then manually terminate the cluster when you no longer need it. The clusters in these examples are referred to as a *long-running clusters*.

Additionally, you can configure termination protection to prevent core instances in your cluster from being terminated due to errors or issues during processing. When termination protection is enabled, you can recover data from instances prior to termination. The default settings for these options differs depending on whether you launch your cluster by using the console, CLI, or API. For more information, see [Control Cluster Termination \(p. 185\)](#).

Security

Amazon EMR leverages other AWS services, such as IAM and Amazon VPC, and features, such as Amazon EC2 key pairs, to help you secure your clusters and data.

IAM

Amazon EMR integrates with IAM to manage permissions. You define permissions using IAM policies, which you attach to IAM users or IAM groups. The permissions that you define in the policy determine the actions that those users or members of the group can perform and the resources that they can access. For more information, see [Configure User Permissions Using IAM Roles \(p. 99\)](#) and [Amazon EMR Actions in User-Based IAM Policies \(p. 100\)](#).

Additionally, Amazon EMR uses IAM roles for the Amazon EMR service itself and the EC2 instance profile for the instances. These roles grant permission for the service and instances to access other AWS services on your behalf. There is a default role for the Amazon EMR service and a default role for the EC2 instance

profile. The default roles use AWS managed policies, which are created for you automatically the first time you launch an EMR cluster from the console and choose default permissions. You can also create the default IAM roles from the AWS CLI. If you want to manage the permissions instead of AWS, you can choose custom roles for the service and instance profile. For more information, see [Configure IAM Roles for Amazon EMR and Applications \(p. 106\)](#).

Security Groups

Amazon EMR uses security groups to control inbound and outbound traffic to your EC2 instances. When you launch your cluster, Amazon EMR uses a security group for your master instance and a security group to be shared by your core/task instances. Amazon EMR configures the security group rules to ensure communication among the instances in the cluster. Optionally, you can configure additional security groups and assign them to your master and core/task instances if you require more advanced rules. For more information, see [Configure Security Groups \(p. 113\)](#).

Encryption

Amazon EMR supports optional Amazon S3 server-side and client-side encryption with EMRFS to help protect the data that you store in Amazon S3. With server-side encryption, Amazon S3 encrypts your data after you upload it.

With client-side encryption, the encryption and decryption process occurs in the EMRFS client on your EMR cluster. You manage the master key for client-side encryption using either the AWS Key Management Service (AWS KMS) or your own key management system.

For more information, see [Encryption for Amazon S3 Data with EMRFS in the Amazon EMR Release Guide](#).

Amazon VPC

Amazon EMR supports launching clusters in a virtual private cloud (VPC) in Amazon VPC. A VPC is an isolated, virtual network in AWS that provides the ability to control advanced aspects of network configuration and access. For more information, see [Plan and Configure Networking \(p. 73\)](#).

AWS CloudTrail

Amazon EMR integrates with CloudTrail to log information about requests made by or on behalf of your AWS account. With this information, you can keep track of who is accessing your cluster when, and the IP address from which they made the request. For more information, see [Logging Amazon EMR API Calls in AWS CloudTrail \(p. 171\)](#).

Amazon EC2 Key Pairs

You can monitor and interact with your cluster by forming a secure connection between your remote computer and the master node. You use the Secure Shell (SSH) network protocol for this connection. When you connect to the master node using SSH, you need the public DNS name of the master node and your private key from the Amazon EC2 key pair that you chose when you created the cluster. This key pair is required to access the EC2 instance of your master node. For more information, see [Create SSH Credentials for the Master Node \(p. 119\)](#).

Monitoring

You can use the Amazon EMR management interfaces and log files to troubleshoot cluster issues, such as failures or errors. Amazon EMR provides the ability to archive log files in Amazon S3 so you can store logs and troubleshoot issues even after your cluster terminates. Amazon EMR also provides an optional debugging tool in the Amazon EMR console to browse the log files based on steps, jobs, and tasks. For more information, see [Configure Cluster Logging and Debugging \(p. 120\)](#).

Amazon EMR integrates with CloudWatch to track performance metrics for the cluster and jobs within the cluster. You can configure alarms based on a variety of metrics such as whether the cluster is idle or the percentage of storage used. For more information, see [Monitor Metrics with CloudWatch \(p. 157\)](#).

Management Interfaces

There are several ways you can interact with Amazon EMR:

- **Console** — A graphical user interface that you can use to launch and manage clusters. With it, you fill out web forms to specify the details of clusters to launch, view the details of existing clusters, debug, and terminate clusters. Using the console is the easiest way to get started with Amazon EMR; no programming knowledge is required. The console is available online at <https://console.aws.amazon.com//elasticmapreduce/home>.
- **AWS Command Line Interface (AWS CLI)** — A client application you run on your local machine to connect to Amazon EMR and create and manage clusters. The AWS CLI contains a feature-rich set of commands specific to Amazon EMR. With it, you can write scripts that automate the process of launching and managing clusters. Using the AWS CLI is the best option if you prefer working from a command line. For more information, see [Amazon EMR](#) in the *AWS Command Line Interface Reference*.
- **Software Development Kit (SDK)** — SDKs provide functions that call Amazon EMR to create and manage clusters. With them, you can write applications that automate the process of creating and managing clusters. Using the SDK is the best option if you want to extend or customize the functionality of Amazon EMR. Amazon EMR is currently available in the following SDKs: Go, Java, .NET (C# and VB.NET), Node.js, PHP, Python, and Ruby. For more information about these SDKs, see [Tools for AWS](#) and [Amazon EMR Sample Code & Libraries](#).
- **Web Service API** — A low-level interface that you can use to call the web service directly, using JSON. Using the API is the best option if you want to create a custom SDK that calls Amazon EMR. For more information, see the [Amazon EMR API Reference](#).

Overview of Amazon EMR Architecture

Amazon EMR service architecture consists of several layers, each of which provides certain capabilities and functionality to the cluster. This section provides an overview of the layers and the components of each.

In This Topic

- [Storage \(p. 9\)](#)
- [Cluster Resource Management \(p. 10\)](#)
- [Data Processing Frameworks \(p. 10\)](#)
- [Applications and Programs \(p. 11\)](#)

Storage

The storage layer includes the different file systems that are used with your cluster. There are several different types of storage options as follows.

Hadoop Distributed File System (HDFS)

Hadoop Distributed File System (HDFS) is a distributed, scalable file system for Hadoop. HDFS distributes the data it stores across instances in the cluster, storing multiple copies of data on different instances to ensure that no data is lost if an individual instance fails. HDFS is ephemeral storage that is reclaimed when you terminate a cluster. HDFS is useful for caching intermediate results during MapReduce processing or for workloads which have significant random I/O.

For more information, go to [HDFS Users Guide](#) on the Apache Hadoop website.

EMR File System (EMRFS)

Using the EMR File System (EMRFS), Amazon EMR extends Hadoop to add the ability to directly access data stored in Amazon S3 as if it were a file system like HDFS. You can use either HDFS or Amazon S3 as the file system in your cluster. Most often, Amazon S3 is used to store input and output data and intermediate results are stored in HDFS.

Local File System

The local file system refers to a locally connected disk. When you create a Hadoop cluster, each node is created from an Amazon EC2 instance that comes with a preconfigured block of preattached disk storage called an instance store. Data on instance store volumes persists only during the lifecycle of its Amazon EC2 instance.

Cluster Resource Management

The resource management layer is responsible for managing cluster resources and scheduling the jobs for processing data.

By default, Amazon EMR uses YARN (Yet Another Resource Negotiator), which is a component introduced in Apache Hadoop 2.0 to centrally manage cluster resources for multiple data-processing frameworks. However, there are other frameworks and applications that are offered in Amazon EMR that do not use YARN as a resource manager. Amazon EMR also has an agent on each node which administers YARN components, keeps the cluster healthy, and communicates with the Amazon EMR service.

Data Processing Frameworks

The data processing framework layer is the engine used to process and analyze data. There are many frameworks available that run on YARN or have their own resource management. Different frameworks are available for different kinds of processing needs, such as batch, interactive, in-memory, streaming, and so on. The framework that you choose depends on your use case. This impacts the languages and interfaces available from the application layer, which is the layer used to interact with the data you want to process. The main processing frameworks available for Amazon EMR are Hadoop MapReduce and Spark.

Hadoop MapReduce

Hadoop MapReduce is an open-source programming model for distributed computing. It simplifies the process of writing parallel distributed applications by handling all of the logic, while you provide the Map and Reduce functions. The Map function maps data to sets of keyvalue pairs called intermediate results. The Reduce function combines the intermediate results, applies additional algorithms, and produces the final output. There are multiple frameworks available for MapReduce, such as Hive, which automatically generate Map and Reduce programs.

For more information, go to [How Map and Reduce operations are actually carried out](#) on the Apache Hadoop Wiki website.

Apache Spark

Spark is a cluster framework and programming model for processing big data workloads. Like Hadoop MapReduce, Spark is an open-source, distributed processing system but uses directed acyclic graphs for execution plans and leverages in-memory caching for datasets. When you run Spark on Amazon EMR, you can use EMRFS to directly access your data in Amazon S3. Spark supports multiple interactive query modules such as SparkSQL.

For more information, see [Apache Spark on Amazon EMR Clusters](#) in the *Amazon EMR Release Guide*.

Applications and Programs

Amazon EMR supports many applications, such as Hive, Pig, and the Spark Streaming library to provide capabilities such as using higher level languages to create processing workloads, leveraging machine learning algorithms, making stream processing applications, and building data warehouses. In addition, Amazon EMR also supports open-source projects that have their own cluster management functionality instead of using YARN.

You use various libraries and languages to interact with the applications that you run in Amazon EMR. For example, you can use Java, Hive, or Pig with MapReduce or Spark Streaming, Spark SQL, MLlib, and GraphX with Spark.

For more information, see the [Amazon EMR Release Guide](#).

Getting Started: Analyzing Big Data with Amazon EMR

This documentation is for versions 4.x and 5.x of Amazon EMR. For information about Amazon EMR AMI versions 2.x and 3.x, see the [Amazon EMR Developer Guide \(PDF\)](#).

This section is a tutorial designed to walk you through the process of creating a sample Amazon EMR cluster by using the AWS Management Console. You then run a Hive script as a step to process sample data that is stored in Amazon S3. This tutorial is not meant for production environments, and it does not cover configuration options in depth. It is meant to help you quickly set up a cluster for evaluation purposes. If you have questions or get stuck, you can reach out to the Amazon EMR team by posting on our [Discussion Forum](#).

The sample cluster that you create will be running in a live environment and you will be charged for the resources that you use. This tutorial should take an hour or less, so the charges that you incur should be minimal. After you complete this tutorial, you should reset your environment to avoid incurring further charges. For more information about resetting your environment, see [Step 5: Reset Your Environment \(p. 23\)](#).

Pricing for Amazon EMR varies by region and service. In this tutorial, charges accrue for the Amazon EMR cluster and Amazon Simple Storage Service (Amazon S3) storage of the log data and output from the Hive job. If you are within your first year of using AWS, some or all of your charges for Amazon S3 might be waived if you are within your usage limits of the AWS Free Tier. For more information about Amazon EMR pricing and the AWS Free Tier, go to [Amazon EMR Pricing](#) and [AWS Free Tier](#).

Note

You can use the [Amazon Web Services Simple Monthly Calculator](#) to estimate your bill.

Steps in This Tutorial

- [Step 1: Set Up Prerequisites for Your Sample Cluster \(p. 13\)](#)
- [Step 2: Launch Your Sample Amazon EMR Cluster \(p. 14\)](#)
- [Step 3: Prepare Your Sample Data and Script \(p. 20\)](#)
- [Step 4: Process Your Sample Data By Running a Hive Script \(p. 22\)](#)

- [Step 5: Reset Your Environment \(p. 23\)](#)

Step 1: Set Up Prerequisites for Your Sample Cluster

Before you begin setting up your Amazon EMR cluster, make sure that you complete the prerequisites in this topic.

In This Topic

- [Sign Up for AWS \(p. 13\)](#)
- [Create an Amazon S3 Bucket \(p. 13\)](#)
- [Create an Amazon EC2 Key Pair \(p. 13\)](#)

Sign Up for AWS

If you do not have an AWS account, use the following procedure to create one.

To sign up for AWS

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

Create an Amazon S3 Bucket

In this tutorial, you use an Amazon S3 bucket to store your log files and output data. Because of Hadoop requirements, S3 bucket names used with Amazon EMR have the following constraints:

- Must contain only lowercase letters, numbers, periods (.), and hyphens (-)
- Cannot end in numbers

If you already have a bucket that meets these requirements, you can use it for this tutorial. Otherwise, create a bucket to use. For more information about creating buckets, go to [Create a Bucket](#) in the *Amazon Simple Storage Service Getting Started Guide*.

In your S3 bucket, create folders named `logs` and `output`. Also, the output folder should be empty. For more information about creating folders, go to [Creating A Folder](#) in the *Amazon Simple Storage Service Console User Guide*.

Create an Amazon EC2 Key Pair

You must have an Amazon Elastic Compute Cloud (Amazon EC2) key pair to connect to the nodes in your cluster over a secure channel using the Secure Shell (SSH) protocol. If you already have a key pair that you want to use, you can skip this step. If you don't have a key pair, follow one of the following procedures depending on your operating system.

- [Creating Your Key Pair Using Amazon EC2](#) in the *Amazon EC2 User Guide for Windows Instances*
- [Creating Your Key Pair Using Amazon EC2](#) in the *Amazon EC2 User Guide for Linux Instances*

Note

Use this procedure for both Linux and Mac OS X operating systems.

Step 2: Launch Your Sample Amazon EMR Cluster

In this step, you launch your sample cluster by using the Amazon EMR console. Before you perform this step, make sure that you meet the requirements in [Step 1: Set Up Prerequisites for Your Sample Cluster](#) (p. 13).

In This Topic

- [Using Quick Cluster Configuration Overview](#) (p. 14)
- [Launch the Sample Cluster](#) (p. 20)

Using Quick Cluster Configuration Overview

The following table describes the fields and default values when you launch a cluster using the **Quick cluster configuration** page in the Amazon EMR console.

Console field	Default value	Description
Cluster name	My cluster	The cluster name is an optional, descriptive name for your cluster that does not need to be unique.
Logging	Enable	This option specifies whether to enable or disable logging. When logging is enabled, Amazon EMR writes detailed log data to a folder in either a S3 bucket chosen for you or your own specified bucket. Logging is an immutable property that can only be enabled when you create the cluster.
S3 folder	s3://aws-logs- <i>account_number</i> - <i>region</i> /elasticmapreduce/	This option specifies the path to a folder in an S3 bucket where you want Amazon EMR to write log data. The following example shows

Console field	Default value	Description
		<p>a path to an S3 bucket for AWS account ID 111122223333 in the us-east-1 region: s3://aws-logs-111122223333-us-east-1/elasticmapreduce/.</p> <p>If the folder in the specified path does not exist in the bucket, it is created for you. You can specify a different folder by typing or browsing to a different location.</p>

Console field	Default value	Description
Launch mode	Cluster	<p>This option specifies whether to launch an ongoing cluster or a transient cluster as follows:</p> <ul style="list-style-type: none">• With the Cluster option, Amazon EMR launches a cluster with the applications that you choose in Software configuration. You can add steps to the cluster after it is launched, and the cluster continues running until you terminate it.• With the Step execution option, you add steps to run after the cluster launches. The steps that you add determine the applications that are included in the cluster. When your cluster launches, and the steps complete, the cluster is automatically terminated.

Console field	Default value	Description
Release	<i>EMR release label</i>	This option specifies the software and Amazon EMR platform components, such as EMRFS, to install on your cluster. Amazon EMR uses the release to initialize the Amazon EC2 instances on which your cluster runs. These releases are specific to Amazon EMR and can be used only in the context of running your Amazon EMR cluster. The latest release label is selected by default.
Applications	All applications (for Cluster launch mode) Core Hadoop (if you choose Step execution launch mode)	This option determines the applications to install on your cluster. If you chose Cluster launch mode, you can select the applications to install. If you chose Step execution launch mode, the list of applications is determined by the steps that you added.
Instance type	m3.xlarge	This option determines the Amazon EC2 instance type that Amazon EMR initializes for the instances that run in your cluster.

Console field	Default value	Description
Number of instances	3	This option determines the number of Amazon EC2 instances to initialize. Each instance corresponds to a node in the Amazon EMR cluster. You must have at least one node.
EC2 key pair	Select an option	This option specifies the Amazon EC2 key pair to use when connecting to the nodes in your cluster using Secure Shell (SSH). If you do not select a key pair, you cannot connect to the cluster.

Console field	Default value	Description
Permissions	Default	<p>This option configures permissions for your Amazon EMR cluster. These permissions are granted using policies that are applied to the following IAM roles:</p> <ul style="list-style-type: none"> • EMR role — Grants Amazon EMR permission to access other AWS services on your behalf. • EC2 instance profile — Grants your cluster's Amazon EC2 instances permission to access other AWS services on your behalf. <p>With Default permissions, the IAM roles use the following AWS managed policies:</p> <p>AmazonElasticMapReduceRole for the Amazon EMR service and AmazonElasticMapReduceforEC2Role for your instance profile. You can choose View policy for EMR role or View policy for EC2 instance profile to view these policies.</p> <p>With Custom permissions, you must select</p>

Console field	Default value	Description
		existing roles. The policies attached to those roles determine the permissions for Amazon EMR and your Amazon EC2 instance profile.

Launch the Sample Cluster

Perform the following steps to launch your sample cluster. Unless otherwise specified in the procedure, use the default values as described in the preceding table.

To launch an Amazon EMR cluster

Sign in to the AWS Management Console and open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.

1. Choose **Create cluster**.
2. On the **Quick cluster configuration** page, accept the default values except for the following fields:
 - For **S3 folder**, choose the folder icon to select the path to the `logs` folder that you created in [Create an Amazon S3 Bucket \(p. 13\)](#).
 - For **EC2 key pair**, choose the key pair that you created in [Create an Amazon EC2 Key Pair \(p. 13\)](#).
3. Choose **Create cluster**.
4. Proceed to the next step.

Step 3: Prepare Your Sample Data and Script

For this tutorial, the sample data and sample script are already provided for you. However, this step describes them so that you understand how they fit into the overall process of using Amazon EMR to analyze data.

In This Topic

- [Sample Data Overview \(p. 20\)](#)
- [Sample Hive Script Overview \(p. 21\)](#)

Sample Data Overview

The sample data is a series of Amazon CloudFront web distribution log files. The data is stored in Amazon S3 at `s3://region.elasticmapreduce.samples` where **region** is your region.

Each entry in the CloudFront log files provides details about a single user request in the following format:

Step 4: Process Your Sample Data By Running a Hive Script

In this step of the tutorial, you run your Hive script in your cluster as a step in the Amazon EMR console to process your sample data. In Amazon EMR, a *step* is a unit of work that contains one or more Hadoop jobs. You can submit steps when you create the cluster or when the cluster is running (if it is a long-running cluster).

In This Topic

- [Submit the Hive Script as a Step \(p. 22\)](#)
- [View the Results \(p. 23\)](#)

Submit the Hive Script as a Step

Use the **Add Step** option to submit your Hive script to the cluster using the console. The Hive script and sample data used by the script have been uploaded to Amazon S3 for you.

Note

Before you run the script, you must have an Amazon S3 bucket and `output` folder as described in [Create an Amazon S3 Bucket \(p. 13\)](#).

To submit your Hive script as a step

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. In **Cluster List**, select the name of your cluster.
3. Scroll to the **Steps** section and expand it, then choose **Add step**.
4. In the **Add Step** dialog:
 - For **Step type**, choose **Hive program**.
 - For **Name**, accept the default name (Hive program) or type a new name.
 - For **Script S3 location**, type `s3://region.elasticmapreduce.samples/cloudfront/code/Hive_CloudFront.q`

Replace *region* with your region. For example, for US West (Oregon) type `s3://us-west-2.elasticmapreduce.samples/cloudfront/code/Hive_CloudFront.q`
 - For **Input S3 location**, type `s3://region.elasticmapreduce.samples`

Replace *region* with your region. For example, for US West (Oregon) type `s3://us-west-2.elasticmapreduce.samples`
 - For **Output S3 location**, type or browse to the `output` bucket that you created in [Create an Amazon S3 Bucket \(p. 13\)](#).
 - For **Arguments**, include the following argument to allow column names that are the same as reserved words:

```
-hiveconf hive.support.sql11.reserved.keywords=false
```
 - For **Action on failure**, accept the default option **Continue**.
5. Choose **Add**. The step appears in the console with a status of **Pending**.
6. The status of the step changes from **Pending** to **Running** to **Completed** as the step runs. To update the status, choose **Refresh** above the **Actions** column. The step runs in approximately 1 minute.

View the Results

After the step completes successfully, the query output produced by the Hive script is stored in the Amazon S3 output folder that you specified when you submitted the step.

To view the output of the Hive script

1. Open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. In the Amazon S3 console, select the bucket that you used for the output data; for example, `s3://myemrbucket/`.
3. Select the `output` folder.
4. The query writes results into a separate folder. Choose `os_requests`.
5. The Hive query results are stored in a text file. To download the file, right-click it, choose **Download**, open the context (right-click) menu for **Download**, choose **Save Link As**, and then save the file to a suitable location.
6. Open the file using a text editor such as WordPad (Windows), TextEdit (Mac OS), or gEdit (Linux). In the output file, you should see the number of access requests by operating system.
7. Proceed to the next step.

Step 5: Reset Your Environment

After you have completed this tutorial, you should remove your Amazon S3 bucket and terminate your Amazon EMR cluster to avoid incurring additional charges.

Deleting Your Amazon S3 Bucket

You cannot delete an Amazon S3 bucket that has items in it. First, delete your `logs` and `output` folders, and then delete your bucket. For more information about deleting folders and buckets, go to [Delete an Object and Bucket](#) in the *Amazon Simple Storage Service Getting Started Guide*.

Terminating Your Sample Cluster

Terminating your cluster terminates the associated Amazon EC2 instances and stops the accrual of Amazon EMR charges. Amazon EMR preserves metadata information about completed clusters for your reference, at no charge, for two months. The console does not provide a way to delete completed clusters from the console; these are automatically removed for you after two months.

To terminate your Amazon EMR cluster

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. On the **Cluster List** page, select your cluster and choose **Terminate**.
3. By default, clusters created using the console are launched with termination protection enabled, so you must disable it. In the **Terminate clusters** dialog, for **Termination protection**, choose **Change**.
4. Choose **Off** and then confirm the change.
5. Choose **Terminate**.

Plan and Configure Clusters

This documentation is for versions 4.x and 5.x of Amazon EMR. For information about Amazon EMR AMI versions 2.x and 3.x, see the [Amazon EMR Developer Guide \(PDF\)](#).

This section explains configuration options and instructions for planning, configuring, and launching clusters using Amazon EMR. Before you launch a cluster, you make choices about your system based on the data that you're processing and your requirements for cost, speed, capacity, availability, security, and manageability. Your choices include:

- What region to run a cluster in, where and how to store data, and how to output results. See [Configure Cluster Location and Data Storage \(p. 24\)](#).
- Whether a cluster is long-running or transient, and what software it runs. See [Configure a Cluster to be Transient or Long-Running \(p. 56\)](#) and [Configure Cluster Software \(p. 63\)](#).
- The hardware and networking options that optimize cost, performance, and availability for your application. See [Configure Cluster Hardware and Networking \(p. 67\)](#).
- How to restrict access to cluster resources and data. See [Configure Access to the Cluster \(p. 99\)](#).
- How to set up clusters so you can manage them more easily, and monitor activity, performance, and health. See [Configure Cluster Logging and Debugging \(p. 120\)](#) and [Tag Clusters \(p. 124\)](#).
- How to integrate with other software and services. See [Drivers and Third-Party Application Integration \(p. 129\)](#).

Configure Cluster Location and Data Storage

This section describes how to configure the region for a cluster, the different file systems available when you use Amazon EMR and how to use them. It also covers how to prepare or upload data to Amazon EMR if necessary, as well as how to prepare an output location for log files and any output data files you configure.

Topics

- [Choose an AWS Region \(p. 25\)](#)
- [Work with Storage and File Systems \(p. 26\)](#)
- [Prepare Input Data \(p. 44\)](#)

- [Configure an Output Location \(p. 52\)](#)

Choose an AWS Region

Amazon Web Services run on servers in data centers around the world. Data centers are organized by geographical region. When you launch an Amazon EMR cluster, you must specify a region. You might choose a region to reduce latency, minimize costs, or address regulatory requirements. For the list of regions and endpoints supported by Amazon EMR, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.

For best performance, you should launch the cluster in the same region as your data. For example, if the Amazon S3 bucket storing your input data is in the US West (Oregon) region, you should launch your cluster in the US West (Oregon) region to avoid cross-region data transfer fees. If you use an Amazon S3 bucket to receive the output of the cluster, you would also want to create it in the US West (Oregon) region.

If you plan to associate an Amazon EC2 key pair with the cluster (required for using SSH to log on to the master node), the key pair must be created in the same region as the cluster. Similarly, the security groups that Amazon EMR creates to manage the cluster are created in the same region as the cluster.

If you signed up for an AWS account on or after May 17, 2017, the default region when you access a resource from the AWS Management Console is US East (Ohio) (us-east-2); for older accounts, the default region is either US West (Oregon) (us-west-2) or US East (N. Virginia) (us-east-1). For more information, see [Regions and Endpoints](#).

Some AWS features are available only in limited regions. For example, Cluster Compute instances are available only in the US East (N. Virginia) region, and the Asia Pacific (Sydney) region supports only Hadoop 1.0.3 and later. When choosing a region, check that it supports the features you want to use.

For best performance, use the same region for all of your AWS resources that will be used with the cluster. The following table maps the region names between services. For a list of Amazon EMR regions, see [AWS Regions and Endpoints](#) in the *Amazon Web Services General Reference*.

Choose a Region Using the Console

Your default region is displayed automatically.

To change regions using the console

- To switch regions, choose the region list to the right of your account information on the navigation bar.

Specify a Region Using the AWS CLI

You specify a default region in the AWS CLI using either the **aws configure** command or the `AWS_DEFAULT_REGION` environment variable. For more information, see [Configuring the AWS Region](#) in the *AWS Command Line Interface User Guide*.

Choose a Region Using an SDK or the API

To choose a region using an SDK, configure your application to use that region's endpoint. If you are creating a client application using an AWS SDK, you can change the client endpoint by calling `setEndpoint`, as shown in the following example:

```
client.setEndpoint("elasticmapreduce.us-west-2.amazonaws.com");
```

After your application has specified a region by setting the endpoint, you can set the Availability Zone for your cluster's EC2 instances. Availability Zones are distinct geographical locations that are engineered to be insulated from failures in other Availability Zones and provide inexpensive, low latency network connectivity to other Availability Zones in the same region. A region contains one or more Availability Zones. To optimize performance and reduce latency, all resources should be located in the same Availability Zone as the cluster that uses them.

Work with Storage and File Systems

Amazon EMR and Hadoop provide a variety of file systems that you can use when processing cluster steps. You specify which file system to use by the prefix of the URI used to access the data. For example, `s3://myawsbucket/path` references an Amazon S3 bucket using EMRFS. The following table lists the available file systems, with recommendations about when it's best to use each one.

Amazon EMR and Hadoop typically use two or more of the following file systems when processing a cluster. HDFS and EMRFS are the two main file systems used with Amazon EMR.

File System	Prefix	Description
HDFS	<code>hdfs://</code> (or no prefix)	<p>HDFS is a distributed, scalable, and portable file system for Hadoop. An advantage of HDFS is data awareness between the Hadoop cluster nodes managing the clusters and the Hadoop cluster nodes managing the individual steps. For more information about how HDFS works, go to the Hadoop documentation.</p> <p>HDFS is used by the master and core nodes. One advantage is that it's fast; a disadvantage is that it's ephemeral storage which is reclaimed when the cluster ends. It's best used for caching the results produced by intermediate job-flow steps.</p>
EMRFS	<code>s3://</code>	<p>EMRFS is an implementation of HDFS used for reading and writing regular files from Amazon EMR directly to Amazon S3. EMRFS provides the convenience of storing persistent data in Amazon S3 for use with Hadoop while also providing features like Amazon S3 server-side encryption, read-after-write consistency, and list consistency.</p> <p>Note Previously, Amazon EMR used the S3 Native FileSystem with the URI scheme, <code>s3n</code>. While this still works, we recommend that you use the <code>s3</code> URI scheme for the best performance, security, and reliability.</p>
local file system		<p>The local file system refers to a locally connected disk. When a Hadoop cluster is created, each node is created from an EC2 instance that comes with a preconfigured block of preattached disk storage called an <i>instance store</i>. Data on instance store volumes persists only during the life of its EC2 instance. Instance store volumes are ideal for storing temporary data that is continually changing, such as buffers, caches, scratch data, and other temporary content. For more information, see Amazon EC2 Instance Storage.</p>

File System	Prefix	Description
(Legacy) Amazon S3 block file system	s3bfs://	<p>The Amazon S3 block file system is a legacy file storage system. We strongly discourage the use of this system.</p> <p>Important We recommend that you do not use this file system because it can trigger a race condition that might cause your cluster to fail. However, it might be required by legacy applications.</p>

Note

The s3a protocol is not supported. We suggest you use s3 in place of s3a and s3n URI prefixes.

Access File Systems

You specify which file system to use by the prefix of the uniform resource identifier (URI) used to access the data. The following procedures illustrate how to reference several different types of file systems.

To access a local HDFS

- Specify the `hdfs:///` prefix in the URI. Amazon EMR resolves paths that do not specify a prefix in the URI to the local HDFS. For example, both of the following URIs would resolve to the same location in HDFS.

```
hdfs:///path-to-data
/path-to-data
```

To access a remote HDFS

- Include the IP address of the master node in the URI, as shown in the following examples.

```
hdfs://master-ip-address/path-to-data
master-ip-address/path-to-data
```

To access Amazon S3

- Use the `s3://` prefix.

```
s3://bucket-name/path-to-file-in-bucket
```

To access the Amazon S3 block file system

- Use only for legacy applications that require the Amazon S3 block file system. To access or store data with this file system, use the `s3bfs://` prefix in the URI.

The Amazon S3 block file system is a legacy file system that was used to support uploads to Amazon S3 that were larger than 5 GB in size. With the multipart upload functionality Amazon EMR provides through the AWS Java SDK, you can upload files of up to 5 TB in size to the Amazon S3 native file system, and the Amazon S3 block file system is deprecated.

Warning

Because this legacy file system can create race conditions that can corrupt the file system, you should avoid this format and use EMRFS instead.

```
s3bfs://bucket-name/path-to-file-in-bucket
```

EMR File System (EMRFS)

The EMR File System (EMRFS) and the Hadoop Distributed File System (HDFS) are both installed as components in the release. EMRFS is an implementation of HDFS which allows clusters to store data on Amazon S3. You can enable Amazon S3 server-side and client-side encryption as well as consistent view for EMRFS using the AWS Management Console, AWS CLI, or you can use a bootstrap action (with CLI or SDK) to configure additional settings for EMRFS.

Enabling Amazon S3 server-side encryption allows you to encrypt objects written to Amazon S3 by EMRFS. EMRFS support for Amazon S3 client-side encryption allows your cluster to work with S3 objects that were previously encrypted using an Amazon S3 encryption client. Consistent view provides consistency checking for list and read-after-write (for new put requests) for objects in Amazon S3. Enabling consistent view requires you to store EMRFS metadata in Amazon DynamoDB. If the metadata is not present, it is created for you.

Topics

- [Consistent View \(p. 28\)](#)
- [Create an AWSCredentialsProvider for EMRFS \(p. 42\)](#)
- [EMRFS Endpoint Resolution \(p. 44\)](#)

Consistent View

EMRFS consistent view monitors Amazon S3 list consistency for objects written by or synced with EMRFS, delete consistency for objects deleted by EMRFS, and read-after-write consistency for new objects written by EMRFS.

Amazon S3 is designed for eventual consistency. For instance, buckets in all regions provide read-after-write consistency for put requests of new objects and eventual consistency for overwrite of put and delete requests. Therefore, if you are listing objects in an Amazon S3 bucket quickly after putting new objects, Amazon S3 does not provide a guarantee to return a consistent listing and it may be incomplete. This is more common in quick sequential MapReduce jobs which use Amazon S3 as a data store.

EMRFS includes a command line utility on the master node, `emrfs`, which allows administrator to perform operations on metadata such as import, delete, and sync. For more information about the EMRFS CLI, see [the section called “EMRFS CLI Reference” \(p. 36\)](#).

For a given path, EMRFS returns the set of objects listed in the EMRFS metadata and those returned directly by Amazon S3. Because Amazon S3 is still the “source of truth” for the objects in a path, EMRFS ensures that everything in a specified Amazon S3 path is being processed regardless of whether it is tracked in the metadata. However, EMRFS consistent view only ensures that the objects in the folders which you are tracking are being checked for consistency. The following topics give further details about how to enable and use consistent view.

Note

If you directly delete objects from Amazon S3 that are being tracked in the EMRFS metadata, EMRFS sees an entry for that object in the metadata but not the object in an Amazon S3 list or get request. Therefore, EMRFS treats the object as inconsistent and throws an exception after it has exhausted retries. You should use EMRFS to delete objects in Amazon S3 that are being tracked in the consistent view, purge the entries in the metadata for objects directly deleted in Amazon S3, or sync the consistent view with Amazon S3 immediately after you delete objects directly from Amazon S3.

To read an article about EMRFS consistency, see the [Ensuring Consistency When Using Amazon S3 and Amazon Elastic MapReduce for ETL Workflows](#) post on the AWS Big Data blog.

Topics

- [Enable Consistent View](#) (p. 29)
- [Objects Tracked By EMRFS](#) (p. 30)
- [Retry Logic](#) (p. 31)
- [EMRFS Metadata](#) (p. 31)
- [Configure Consistency Notifications for CloudWatch and Amazon SQS](#) (p. 33)
- [Configure Consistent View](#) (p. 34)
- [EMRFS CLI Reference](#) (p. 36)

Enable Consistent View

You can enable Amazon S3 server-side encryption or consistent view for EMRFS using the AWS Management Console, AWS CLI, or the `emrfs-site` configuration classification.

To configure consistent view using the console

1. Choose **Create Cluster**.
2. Navigate to the **File System Configuration** section.
3. To enable **Consistent view**, choose **Enabled**.
4. For **EMRFS Metadata store**, type the name of your metadata store. The default value is `EmrFSMetadata`. If the `EmrFSMetadata` table does not exist, it is created for you in DynamoDB.

Note

Amazon EMR does not automatically remove the EMRFS metadata from DynamoDB when the cluster is terminated.

5. For **Number of retries**, type an integer value. This value represents the number of times EMRFS retries calling Amazon S3 if an inconsistency is detected. The default value is 5.
6. For **Retry period (in seconds)**, type an integer value. This value represents the amount of time that lapses before EMRFS retries calling Amazon S3. The default value is 10.

Note

Subsequent retries use an exponential backoff.

To launch a cluster with consistent view enabled using the AWS CLI

We recommend you install the current version of AWS CLI. To download the latest release, see <https://aws.amazon.com/cli/>.

- **Note**
Linux line continuation characters (`\`) are included for readability. They can be removed or used in Linux commands. For Windows, remove them or replace with a caret (^).

```
aws emr create-cluster --instance-type m1.large --instance-count 3 --emrfs  
Consistent=true \  
--release-label emr-4.1.0emr-4.2.0 --ec2-attributes KeyName=myKey
```

To check if consistent view is enabled using the AWS Management Console

- To check whether consistent view is enabled in the console, navigate to the **Cluster List** and select your cluster name to view **Cluster Details**. The "EMRFS consistent view" field has a value of `Enabled` or `Disabled`.

To check if consistent view is enabled by examining the `emrfs-site.xml` file

- You can check if consistency is enabled by inspecting the `emrfs-site.xml` configuration file on the master node of the cluster. If the Boolean value for `fs.s3.consistent` is set to `true` then consistent view is enabled for file system operations involving Amazon S3.

Objects Tracked By EMRFS

EMRFS creates a consistent view of objects in Amazon S3 by adding information about those objects to the EMRFS metadata. EMRFS adds these listings to its metadata when:

- An object written by EMRFS during the course of an Amazon EMR job.
- An object is synced with or imported to EMRFS metadata by using the EMRFS CLI.

Objects read by EMRFS are not automatically added to the metadata. When a object is deleted by EMRFS, a listing still remains in the metadata with a deleted state until that listing is purged using the EMRFS CLI. To learn more about the CLI, see [the section called "EMRFS CLI Reference" \(p. 36\)](#). For more information about purging listings in the EMRFS metadata, see [the section called "EMRFS Metadata" \(p. 31\)](#).

For every Amazon S3 operation, EMRFS checks the metadata for information about the set of objects in consistent view. If EMRFS finds that Amazon S3 is inconsistent during one of these operations, it will retry the operation according to parameters defined in `emrfs-site.xml`. After retries are exhausted, it will either throw a `ConsistencyException` or log the exception and continue the workflow. For more information about this retry logic, see [\(p. 31\)](#). You can find `ConsistencyExceptions` in your logs, for example:

- `listStatus: No s3 object for metadata item /s3_bucket/dir/object`
- `getFileStatus: Key dir/file is present in metadata but not s3`

If you delete an object that is being tracked in the EMRFS consistent view directly from Amazon S3, EMRFS will treat that object as inconsistent because it will still be listed in the metadata as present in Amazon S3. If your metadata becomes out of sync with the objects it is tracking in Amazon S3, you can use the **sync** subcommand on the EMRFS CLI to reset the listings in the metadata to reflect what is currently in Amazon S3. To find if there is a discrepancy between the metadata and Amazon S3, you can use the **diff** subcommand on the EMRFS CLI to compare them. Finally, EMRFS only has a consistent view of the objects referenced in the metadata; there can be other objects in the same Amazon S3 path that are not being tracked. When EMRFS lists the objects in an Amazon S3 path, it will return the superset of the objects being tracked in the metadata and those in that Amazon S3 path.

Retry Logic

EMRFS will try to verify list consistency for objects tracked in its metadata for a specific number of retries. The default is 5. In the case where the number of retries is exceeded the originating job returns a failure unless `fs.s3.consistent.throwExceptionOnInconsistency` is set to `false`, where it will only log the objects tracked as inconsistent. EMRFS uses an exponential backoff retry policy by default but you can also set it to a fixed policy. Users may also want to retry for a certain period of time before proceeding with the rest of their job without throwing an exception. They can achieve this by setting `fs.s3.consistent.throwExceptionOnInconsistency` to `false`, `fs.s3.consistent.retryPolicyType` to `fixed`, and `fs.s3.consistent.retryPeriodSeconds` for the desired value. The following example will create a cluster with consistency enabled, which will log inconsistencies and set a fixed retry interval of 10 seconds:

Example Setting retry period to a fixed amount

```
aws emr create-cluster --release-label emr-4.1.0emr-4.2.0 \  
--instance-type m3.xlarge --instance-count 1 \  
--emrfs Consistent=true,Args=[fs.s3.consistent.throwExceptionOnInconsistency=false,  
fs.s3.consistent.retryPolicyType=fixed,fs.s3.consistent.retryPeriodSeconds=10] --ec2-  
attributes KeyName=myKey
```

Note

Linux line continuation characters (`\`) are included for readability. They can be removed or used in Linux commands. For Windows, remove them or replace with a caret (^).

For more information, see [Consistent View](#) (p. 28).

EMRFS Metadata

Note

In order to use consistent view, your data is tracked in a DynamoDB database. Therefore, you will incur the cost of using that database while it exists.

Amazon EMR tracks consistency using a DynamoDB table to store object state. EMRFS consistent view creates and uses EMRFS metadata stored in a DynamoDB table to maintain a consistent view of Amazon S3 and this consistent view can be shared by multiple clusters. EMRFS creates and uses this metadata to track objects in Amazon S3 folders which have been synced with or created by EMRFS. The metadata is used to track all operations (read, write, update, and copy), and no actual content is stored in it. This metadata is used to validate whether the objects or metadata received from Amazon S3 matches what is expected. This confirmation gives EMRFS the ability to check list consistency and read-after-write consistency for new objects EMRFS writes to Amazon S3 or objects synced with EMRFS.

How to add entries to metadata

You can use the `sync` or `import` subcommands to add entries to metadata. `sync` will simply reflect the state of the Amazon S3 objects in a path while `import` is used strictly to add new entries to the metadata. For more information, see [the section called "EMRFS CLI Reference"](#) (p. 36).

How to check differences between metadata and objects in Amazon S3

To check for differences between the metadata and Amazon S3, use the `diff` subcommand of the EMRFS CLI. For more information, see [the section called "EMRFS CLI Reference"](#) (p. 36).

How to know if metadata operations are being throttled

EMRFS sets default throughput capacity limits on the metadata for its read and write operations at 400 and 100 units, respectively. Large numbers of objects or buckets may cause operations to exceed this capacity, at which point they will be throttled by DynamoDB. For example, an application may cause

EMRFS to throw a `ProvisionedThroughputExceededException` if you are performing an operation that exceeds these capacity limits. Upon throttling the EMRFS CLI tool will attempt to retry writing to the DynamoDB table using [exponential backoff](#) until the operation finishes or when it reaches the maximum retry value for writing objects from EMR to Amazon S3.

You can also view Amazon CloudWatch metrics for your EMRFS metadata in the DynamoDB console where you can see the number of throttled read and/or write requests. If you do have a non-zero value for throttled requests, your application may potentially benefit from increasing allocated throughput capacity for read or write operations. You may also realize a performance benefit if you see that your operations are approaching the maximum allocated throughput capacity in reads or writes for an extended period of time.

Throughput characteristics for notable EMRFS operations

The default for read and write operations is 400 and 100 throughput capacity units, respectively. The following performance characteristics will give you an idea of what throughput is required for certain operations. These tests were performed using a single-node `m3.large` cluster. All operations were single threaded. Performance will differ greatly based on particular application characteristics and it may take experimentation to optimize file system operations.

Operation	Average read-per-second	Average write-per-second
create (object)	26.79	6.70
delete (object)	10.79	10.79
delete (directory containing 1000 objects)	21.79	338.40
getFileStatus (object)	34.70	0
getFileStatus (directory)	19.96	0
listStatus (directory containing 1 object)	43.31	0
listStatus (directory containing 10 objects)	44.34	0
listStatus (directory containing 100 objects)	84.44	0
listStatus (directory containing 1,000 objects)	308.81	0
listStatus (directory containing 10,000 objects)	416.05	0
listStatus (directory containing 100,000 objects)	823.56	0
listStatus (directory containing 1M objects)	882.36	0
mkdir (continuous for 120 seconds)	24.18	4.03
mkdir	12.59	0
rename (object)	19.53	4.88

Operation	Average read-per-second	Average write-per-second
rename (directory containing 1000 objects)	23.22	339.34

To submit a step that purges old data from your metadata store

Users may wish to remove particular entries in the DynamoDB-based metadata. This can help reduce storage costs associated with the table. Users have the ability to manually or programmatically purge particular entries by using the EMRFS CLI `delete` subcommand. However, if you delete entries from the metadata, EMRFS no longer makes any checks for consistency.

Programmatically purging after the completion of a job can be done by submitting a final step to your cluster which executes a command on the EMRFS CLI. For instance, type the following command to submit a step to your cluster to delete all entries older than two days.

```
aws emr add-steps --cluster-id j-2AL4XXXXXX5T9 --steps Name="emrfsCLI",Jar="command-
runner.jar",Args=["emrfs","delete","--time","2","time-unit","days"]
{
  "StepIds": [
    "s-B12345678902"
  ]
}
```

Use the StepId value returned to check the logs for the result of the operation.

Configure Consistency Notifications for CloudWatch and Amazon SQS

You can enable CloudWatch metrics and Amazon SQS messages in EMRFS for Amazon S3 eventual consistency issues.

CloudWatch

When CloudWatch metrics are enabled, a metric named **Inconsistency** is pushed each time a `FileSystem` API call fails due to Amazon S3 eventual consistency.

To view CloudWatch metrics for Amazon S3 eventual consistency issues

To view the **Inconsistency** metric in the CloudWatch console, select the EMRFS metrics and then select a **JobFlowId/Metric Name** pair. For example: `j-162XXXXXXM2CU ListStatus`, `j-162XXXXXXM2CU GetFileStatus`, and so on.

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the **Dashboard**, in the **Metrics** section, choose **EMRFS**.
3. In the **Job Flow Metrics** pane, select one or more **JobFlowId/Metric Name** pairs. A graphical representation of the metrics appears in the window below.

Amazon SQS

When Amazon SQS notifications are enabled, an Amazon SQS queue with the name `EMRFS-Inconsistency-<jobFlowId>` is created when EMRFS is initialized. Amazon SQS messages are pushed into the queue when a `FileSystem` API call fails due to Amazon S3 eventual consistency. The message contains information such as JobFlowId, API, a list of inconsistent paths, a stack trace, and so on. Messages can be read using the Amazon SQS console or using the EMRFS `read-sqs` command.

To manage Amazon SQS messages for Amazon S3 eventual consistency issues

Amazon SQS messages for Amazon S3 eventual consistency issues can be read using the EMRFS CLI. To read messages from an EMRFS Amazon SQS queue, type the `read-sqs` command and specify an output location on the master node's local file system for the resulting output file.

You can also delete an EMRFS Amazon SQS queue using the `delete-sqs` command.

1. To read messages from an Amazon SQS queue, type the following command. Replace `queueName` with the name of the Amazon SQS queue that you configured and replace `/path/filename` with the path to the output file:

```
emrfs read-sqs -queue-name queueName -output-file /path/filename
```

For example, to read and output Amazon SQS messages from the default queue, type:

```
emrfs read-sqs -queue-name EMRFS-Inconsistency-j-162XXXXXXM2CU -output-file /path/filename
```

Note

You can also use the `-q` and `-o` shortcuts instead of `-queue-name` and `-output-file` respectively.

2. To delete an Amazon SQS queue, type the following command:

```
emrfs delete-sqs -queue-name queueName
```

For example, to delete the default queue, type:

```
emrfs delete-sqs -queue-name EMRFS-Inconsistency-j-162XXXXXXM2CU
```

Note

You can also use the `-q` shortcut instead of `-queue-name`.

Configure Consistent View

You can configure additional settings for consistent view by providing them for the `/home/hadoop/conf/emrfs-site.xml` file by either using AWS CLI or a bootstrap action. For example, you can choose a different default DynamoDB throughput by supplying the following arguments to the CLI `--emrfs` option or bootstrap action:

Example Changing default metadata read and write values at cluster launch

```
aws emr create-cluster --release-label emr-4.1.0emr-4.2.0 --instance-type m3.xlarge \
--emrfs Consistent=true,Args=[fs.s3.consistent.metadata.read.capacity=600,\
fs.s3.consistent.metadata.write.capacity=300] --ec2-attributes KeyName=myKey
```

Alternatively, use the following configuration file and save it locally or in Amazon S3:

```
[
  {
    "Classification": "emrfs-site",
    "Properties": {
      "fs.s3.consistent.metadata.read.capacity": "600",
      "fs.s3.consistent.metadata.write.capacity": "300"
    }
  }
]
```

```
}
}
]
```

Use the configuration you created with the following syntax:

```
aws emr create-cluster --release-label emr-4.1.0emr-4.2.0 --applications Name=Hive \
--instance-type m3.xlarge --instance-count 2 --configurations file:///myConfig.json
```

Note

Linux line continuation characters (\) are included for readability. They can be removed or used in Linux commands. For Windows, remove them or replace with a caret (^).

The following options can be set using configurations or AWS CLI `--emrfs` arguments. For information about those arguments, see the [AWS Command Line Interface Reference](#).

`emrfs-site.xml` properties for consistent view

Property	Default value	Description
<code>fs.s3.consistent</code>	false	When set to true , this property configures EMRFS to use DynamoDB to provide consistency.
<code>fs.s3.consistent.retryPolicyType</code>	exponential	This property identifies the policy to use when retrying for consistency issues. Options include: exponential, fixed, or none.
<code>fs.s3.consistent.retryPeriodSeconds</code>	10	This property sets the length of time to wait between consistency retry attempts.
<code>fs.s3.consistent.retryCount</code>	5	This property sets the maximum number of retries when inconsistency is detected.
<code>fs.s3.consistent.throwExceptionOnInconsistency</code>	true	This property determines whether to throw or log a consistency exception. When set to true , a <code>ConsistencyException</code> is thrown.
<code>fs.s3.consistent.metadata.autoCreate</code>	true	When set to true , this property enables automatic creation of metadata tables.
<code>fs.s3.consistent.metadata.tableName</code>	EmrFSMetadata	This property specifies the name of the metadata table in DynamoDB.
<code>fs.s3.consistent.metadata.read.capacity</code>	400	This property specifies the DynamoDB read capacity to provision when the metadata table is created.
<code>fs.s3.consistent.metadata.write.capacity</code>	100	This property specifies the DynamoDB write capacity to provision when the metadata table is created.

Property	Default value	Description
<code>fs.s3.consistent.fastList</code>	true	When set to true , this property uses multiple threads to list a directory (when necessary). Consistency must be enabled in order to use this property.
<code>fs.s3.consistent.fastList.prefetchMetadata</code>	false	When set to true , this property enables metadata prefetching for directories containing more than 20,000 items.
<code>fs.s3.consistent.notification.CloudWatch</code>	false	When set to true , CloudWatch metrics are enabled for FileSystem API calls that fail due to Amazon S3 eventual consistency issues.
<code>fs.s3.consistent.notification.SQS</code>	false	When set to true , eventual consistency notifications are pushed to an Amazon SQS queue.
<code>fs.s3.consistent.notification.SQS.queueName</code>	EMRFS-Inconsistency- <jobFlowId>	Changing this property allows you to specify your own SQS queue name for messages regarding Amazon S3 eventual consistency issues.
<code>fs.s3.consistent.notification.SQS.customMsg</code>	none	This property allows you to specify custom information included in SQS messages regarding Amazon S3 eventual consistency issues. If a value is not specified for this property, the corresponding field in the message is empty.
<code>fs.s3.consistent.dynamodb.endpoint</code>	none	This property allows you to specify a custom DynamoDB endpoint for your consistent view metadata.

EMRFS CLI Reference

The EMRFS CLI is installed by default on all cluster master nodes created using AMI 3.2.1 or greater. You use the EMRFS CLI to manage the metadata, which tracks when objects have a consistent view.

Note

The **emrfs** command is only supported with VT100 terminal emulation. However, it may work with other terminal emulator modes.

emrfs top-level command

The **emrfs** top-level command supports the following structure.

```
emrfs [describe-metadata | set-metadata-capacity | delete-metadata | create-metadata | \
list-metadata-stores | diff | delete | sync | import ] [options] [arguments]
```

Specify [options], with or without [arguments] as described in the following table. For [options] specific to sub-commands (describe-metadata, set-metadata-capacity, etc.), see each sub-command below.

[options]for emrfs

Option	Description	Required
<code>-a <i>AWS_ACCESS_KEY_ID</i> --access-key <i>AWS_ACCESS_KEY_ID</i></code>	The AWS access key you use to write objects to Amazon S3 and to create or access a metadata store in DynamoDB. By default, <i>AWS_ACCESS_KEY_ID</i> is set to the access key used to create the cluster.	No
<code>-s <i>AWS_SECRET_ACCESS_KEY</i> --secret-key <i>AWS_SECRET_ACCESS_KEY</i></code>	The AWS secret key associated with the access key you use to write objects to Amazon S3 and to create or access a metadata store in DynamoDB. By default, <i>AWS_SECRET_ACCESS_KEY</i> is set to the secret key associated with the access key used to create the cluster.	No
<code>-v --verbose</code>	Makes output verbose.	No
<code>-h --help</code>	Displays the help message for the <code>emrfs</code> command with a usage statement.	No

emrfs describe-metadata sub-command

[options] for emrfs describe-metadata

Option	Description	Required
<code>-m <i>METADATA_NAME</i> --metadata-name <i>METADATA_NAME</i></code>	<i>METADATA_NAME</i> is the name of the DynamoDB metadata table. If the <i>METADATA_NAME</i> argument is not supplied, the default value is <code>EmrFSMetadata</code> .	No

emrfs describe-metadata example

The following example describes the default metadata table.

```
$ emrfs describe-metadata
EmrFSMetadata
  read-capacity: 400
  write-capacity: 100
  status: ACTIVE
  approximate-item-count (6 hour delay): 12
```

emrfs set-metadata-capacity sub-command

[options] for emrfs set-metadata-capacity

Option	Description	Required
<code>-m <i>METADATA_NAME</i> --metadata-name <i>METADATA_NAME</i></code>	<i>METADATA_NAME</i> is the name of the DynamoDB metadata table. If the <i>METADATA_NAME</i> argument is not supplied, the default value is <code>EmrFSMetadata</code> .	No
<code>-r <i>READ_CAPACITY</i> --read-capacity <i>READ_CAPACITY</i></code>	The requested read throughput capacity for the metadata table. If the <i>READ_CAPACITY</i> argument is not supplied, the default value is 400.	No

Option	Description	Required
<code>-w <i>WRITE_CAPACITY</i> --write-capacity <i>WRITE_CAPACITY</i></code>	The requested write throughput capacity for the metadata table. If the <i>WRITE_CAPACITY</i> argument is not supplied, the default value is 100.	No

emrfs set-metadata-capacity example

The following example sets the read throughput capacity to 600 and the write capacity to 150 for a metadata table named `EmrMetadataAlt`.

```
$ emrfs set-metadata-capacity --metadata-name EmrMetadataAlt --read-capacity 600 --write-capacity 150
  read-capacity: 400
  write-capacity: 100
  status: UPDATING
  approximate-item-count (6 hour delay): 0
```

emrfs delete-metadata sub-command

[options] for emrfs delete-metadata

Option	Description	Required
<code>-m <i>METADATA_NAME</i> --metadata-name <i>METADATA_NAME</i></code>	<i>METADATA_NAME</i> is the name of the DynamoDB metadata table. If the <i>METADATA_NAME</i> argument is not supplied, the default value is <code>EmrFSMetadata</code> .	No

emrfs delete-metadata example

The following example deletes the default metadata table.

```
$ emrfs delete-metadata
```

emrfs create-metadata sub-command

[options] for emrfs create-metadata

Option	Description	Required
<code>-m <i>METADATA_NAME</i> --metadata-name <i>METADATA_NAME</i></code>	<i>METADATA_NAME</i> is the name of the DynamoDB metadata table. If the <i>METADATA_NAME</i> argument is not supplied, the default value is <code>EmrFSMetadata</code> .	No
<code>-r <i>READ_CAPACITY</i> --read-capacity <i>READ_CAPACITY</i></code>	The requested read throughput capacity for the metadata table. If the <i>READ_CAPACITY</i> argument is not supplied, the default value is 400.	No
<code>-w <i>WRITE_CAPACITY</i> --write-capacity <i>WRITE_CAPACITY</i></code>	The requested write throughput capacity for the metadata table. If the <i>WRITE_CAPACITY</i> argument is not supplied, the default value is 100.	No

emrfs create-metadata example

The following example creates a metadata table named `EmrFSMetadataAlt`.

```
$ emrfs create-metadata -m EmrFSMetadataAlt
Creating metadata: EmrFSMetadataAlt
EmrFSMetadataAlt
  read-capacity: 400
  write-capacity: 100
  status: ACTIVE
  approximate-item-count (6 hour delay): 0
```

emrfs list-metadata-stores sub-command

The **emrfs list-metadata-stores** sub-command has no [options].

list-metadata-stores example

The following example lists your metadata tables.

```
$ emrfs list-metadata-stores
EmrFSMetadata
```

emrfs diff sub-command

[options] for emrfs diff

Option	Description	Required
-m <i>METADATA_NAME</i> --metadata-name <i>METADATA_NAME</i>	<i>METADATA_NAME</i> is the name of the DynamoDB metadata table. If the <i>METADATA_NAME</i> argument is not supplied, the default value is <code>EmrFSMetadata</code> .	No
<i>s3://s3Path</i>	The path to the Amazon S3 bucket you are tracking for consistent view that you wish to compare to the metadata table. Buckets sync recursively.	Yes

emrfs diff example

The following example compares the default metadata table to an Amazon S3 bucket.

```
$ emrfs diff s3://elasticmapreduce/samples/cloudfront
BOTH | MANIFEST ONLY | S3 ONLY
DIR elasticmapreduce/samples/cloudfront
DIR elasticmapreduce/samples/cloudfront/code/
DIR elasticmapreduce/samples/cloudfront/input/
DIR elasticmapreduce/samples/cloudfront/logprocessor.jar
DIR elasticmapreduce/samples/cloudfront/input/XABCD12345678.2009-05-05-14.WxYz1234
DIR elasticmapreduce/samples/cloudfront/input/XABCD12345678.2009-05-05-15.WxYz1234
DIR elasticmapreduce/samples/cloudfront/input/XABCD12345678.2009-05-05-16.WxYz1234
DIR elasticmapreduce/samples/cloudfront/input/XABCD12345678.2009-05-05-17.WxYz1234
DIR elasticmapreduce/samples/cloudfront/input/XABCD12345678.2009-05-05-18.WxYz1234
DIR elasticmapreduce/samples/cloudfront/input/XABCD12345678.2009-05-05-19.WxYz1234
DIR elasticmapreduce/samples/cloudfront/input/XABCD12345678.2009-05-05-20.WxYz1234
DIR elasticmapreduce/samples/cloudfront/code/cloudfront-loganalyzer.tgz
```

emrfs delete sub-command

[options] for emrfs delete

Option	Description	Required
<code>-m <i>METADATA_NAME</i> --metadata-name <i>METADATA_NAME</i></code>	<i>METADATA_NAME</i> is the name of the DynamoDB metadata table. If the <i>METADATA_NAME</i> argument is not supplied, the default value is <code>EmrFSMetadata</code> .	No
<code>s3://s3Path</code>	The path to the Amazon S3 bucket you are tracking for consistent view. Buckets sync recursively.	Yes
<code>-t <i>TIME</i> --time <i>TIME</i></code>	The expiration time (interpreted using the time unit argument). All metadata entries older than the <i>TIME</i> argument are deleted for the specified bucket.	
<code>-u <i>UNIT</i> --time-unit <i>UNIT</i></code>	The measure used to interpret the time argument (nanoseconds, microseconds, milliseconds, seconds, minutes, hours, or days). If no argument is specified, the default value is days.	
<code>--read-consumption <i>READ_CONSUMPTION</i></code>	The requested amount of available read throughput used for the delete operation. If the <i>READ_CONSUMPTION</i> argument is not specified, the default value is 400.	No
<code>--write-consumption <i>WRITE_CONSUMPTION</i></code>	The requested amount of available write throughput used for the delete operation. If the <i>WRITE_CONSUMPTION</i> argument is not specified, the default value is 100.	No

emrfs delete example

The following example removes all objects in an Amazon S3 bucket from the tracking metadata for consistent view.

```
$ emrfs delete s3://elasticmapreduce/samples/cloudfront
entries deleted: 11
```

emrfs import sub-command

[options] for emrfs import

Option	Description	Required
<code>-m <i>METADATA_NAME</i> --metadata-name <i>METADATA_NAME</i></code>	<i>METADATA_NAME</i> is the name of the DynamoDB metadata table. If the <i>METADATA_NAME</i> argument is not supplied, the default value is <code>EmrFSMetadata</code> .	No
<code>s3://s3Path</code>	The path to the Amazon S3 bucket you are tracking for consistent view. Buckets sync recursively.	Yes
<code>--read-consumption <i>READ_CONSUMPTION</i></code>	The requested amount of available read throughput used for the delete operation. If the <i>READ_CONSUMPTION</i> argument is not specified, the default value is 400.	No
<code>--write-consumption <i>WRITE_CONSUMPTION</i></code>	The requested amount of available write throughput used for the delete operation. If the	No

Option	Description	Required
	<code>WRITE_CONSUMPTION</code> argument is not specified, the default value is 100.	

emrfs import example

The following example imports all objects in an Amazon S3 bucket with the tracking metadata for consistent view. All unknown keys are ignored.

```
$ emrfs import s3://elasticmapreduce/samples/cloudfront
```

emrfs sync sub-command

[options] for emrfs sync

Option	Description	Required
<code>-m METADATA_NAME</code> <code>--metadata-name METADATA_NAME</code>	<code>METADATA_NAME</code> is the name of the DynamoDB metadata table. If the <code>METADATA_NAME</code> argument is not supplied, the default value is <code>EmrFSMetadata</code> .	No
<code>s3://s3Path</code>	The path to the Amazon S3 bucket you are tracking for consistent view. Buckets sync recursively.	Yes
<code>--read-consumption READ_CONSUMPTION</code>	The requested amount of available read throughput used for the delete operation. If the <code>READ_CONSUMPTION</code> argument is not specified, the default value is 400.	No
<code>--write-consumption WRITE_CONSUMPTION</code>	The requested amount of available write throughput used for the delete operation. If the <code>WRITE_CONSUMPTION</code> argument is not specified, the default value is 100.	No

emrfs sync command example

The following example imports all objects in an Amazon S3 bucket with the tracking metadata for consistent view. All unknown keys are deleted.

```
$ emrfs sync s3://elasticmapreduce/samples/cloudfront
Synching samples/cloudfront                0 added | 0 updated | 0
  removed | 0 unchanged
Synching samples/cloudfront/code/           1 added | 0 updated | 0
  removed | 0 unchanged
Synching samples/cloudfront/                2 added | 0 updated | 0
  removed | 0 unchanged
Synching samples/cloudfront/input/          9 added | 0 updated | 0
  removed | 0 unchanged
Done synching s3://elasticmapreduce/samples/cloudfront 9 added | 0 updated | 1
  removed | 0 unchanged
creating 3 folder key(s)
folders written: 3
```

emrfs read-sqs sub-command

[options] for emrfs read-sqs

Option	Description	Required
<code>-q <i>QUEUE_NAME</i> --queue-name <i>QUEUE_NAME</i></code>	<i>QUEUE_NAME</i> is the name of the Amazon SQS queue configured in <code>emrfs-site.xml</code> . The default value is <code>EMRFS-Inconsistency-<jobFlowId></code> .	Yes
<code>-o <i>OUTPUT_FILE</i> --output-file <i>OUTPUT_FILE</i></code>	<i>OUTPUT_FILE</i> is the path to the output file on the master node's local file system. Messages read from the queue are written to this file.	Yes

emrfs delete-sqs sub-command

[options] for emrfs delete-sqs

Option	Description	Required
<code>-q <i>QUEUE_NAME</i> --queue-name <i>QUEUE_NAME</i></code>	<i>QUEUE_NAME</i> is the name of the Amazon SQS queue configured in <code>emrfs-site.xml</code> . The default value is <code>EMRFS-Inconsistency-<jobFlowId></code> .	Yes

Submitting EMRFS CLI Commands as Steps

The following example shows how to use the `emrfs` utility on the master node by leveraging the AWS CLI or API and the `script-runner.jar` to run the `emrfs` command as a step. The example uses the AWS SDK for Python (Boto) to add a step to a cluster which adds objects in an Amazon S3 bucket to the default EMRFS metadata table.

```
from boto.emr import EmrConnection, connect_to_region, JarStep

emr=EmrConnection()
connect_to_region("us-east-1")

myStep = JarStep(name='Boto EMRFS Sync',
                 jar='s3://elasticmapreduce/libs/script-runner/script-runner.jar',
                 action_on_failure="CONTINUE",
                 step_args=['/home/hadoop/bin/emrfs',
                           'sync',
                           's3://elasticmapreduce/samples/cloudfront'])

stepId = emr.add_jobflow_steps("j-2AL4XXXXXX5T9",
                              steps=[myStep]).stepids[0].value
```

You can use the `stepId` value returned to check the logs for the result of the operation.

Create an AWSCredentialsProvider for EMRFS

You can create a custom credentials provider which implements both the [AWSCredentialsProvider](#) and the Hadoop [Configurable](#) classes for use with EMRFS when it makes calls to Amazon S3. Creating a custom credentials provider in this way is useful when an Amazon EMR user may need to provide different credentials depending on which Amazon S3 bucket is being accessed.

You must specify the full class name of the provider by setting `fs.s3.customAWSCredentialsProvider` in the `emrfs-site` configuration classification. You set this property at cluster creation time using

the AWS CLI. For example, the following code sets `fs.s3.customAWSCredentialsProvider` to *MyAWSCredentialsProvider*.

Use the following configuration file and save it locally or in Amazon S3:

```
[
  {
    "Classification": "emrfs-site",
    "Properties": {
      "fs.s3.customAWSCredentialsProvider": "MyAWSCredentialsProvider"
    }
  }
]
```

Use the configuration you created with the following syntax:

```
aws emr create-cluster --release-label emr-4.1.0emr-4.2.0 --applications Name=Hive \
--instance-type m3.xlarge --instance-count 2 --configurations file:///myConfig.json
```

An example implementation follows:

```
public class MyAWSCredentialsProvider implements AWSCredentialsProvider, Configurable {

    private Configuration conf;
    private String accessKey;
    private String secretKey;

    private void init() {
        accessKey = conf.get("my.accessKey");
        secretKey = conf.get("my.secretKey");
    }

    @Override
    public AWSCredentials getCredentials() {
        return new BasicAWSCredentials(accessKey, secretKey);
    }

    @Override
    public void refresh() {
    }

    @Override
    public void setConf(Configuration configuration) {
        this.conf = configuration;
        init();
    }

    @Override
    public Configuration getConf() {
        return this.conf;
    }
}
```

An alternative implementation allows you to provide the bucket URI when creating an object. That follows this signature:

```
class MyCustomCredentialProvider implements AWSCredentialsProvider , Configurable {
    public MyCustomCredentialProvider (Uri uri , Configuration configuration) {
    }
}
```



```
}
```

EMRFS Endpoint Resolution

EMRFS now resolves to the regional endpoints in which a cluster is running when it makes calls to Amazon S3. Cross-region calls between clusters and buckets not co-located in the same region may fail. This may affect scripts and other artifacts that use older non-regionalized URIs. For more information about region-specific Amazon S3 endpoints, see [AWS Regions and Endpoints](#) in the Amazon Web Services General Reference.

Prepare Input Data

Most clusters load input data and then process that data. In order to load data, it needs to be in a location that the cluster can access and in a format the cluster can process. The most common scenario is to upload input data into Amazon S3. Amazon EMR provides tools for your cluster to import or read data from Amazon S3.

The default input format in Hadoop is text files, though you can customize Hadoop and use tools to import data stored in other formats.

Topics

- [Types of Input Amazon EMR Can Accept](#) (p. 44)
- [How to Get Data Into Amazon EMR](#) (p. 44)

Types of Input Amazon EMR Can Accept

The default input format for a cluster is text files with each line separated by a newline (\n) character. This is the input format most commonly used.

If your input data is in a format other than the default text files, you can use the Hadoop interface `InputFormat` to specify other input types. You can even create a subclass of the `FileInputFormat` class to handle custom data types. For more information, go to <http://hadoop.apache.org/docs/current/api/org/apache/hadoop/mapred/InputFormat.html>.

If you are using Hive, you can use a serializer/deserializer (SerDe) to read data in from a given format into HDFS. For more information, see <https://cwiki.apache.org/confluence/display/Hive/SerDe>.

How to Get Data Into Amazon EMR

Amazon EMR provides several ways to get data onto a cluster. The most common way is to upload the data to Amazon S3 and use the built-in features of Amazon EMR to load the data onto your cluster. You can also use the Distributed Cache feature of Hadoop to transfer files from a distributed file system to the local file system. The implementation of Hive provided by Amazon EMR (version 0.7.1.1 and later) includes functionality that you can use to import and export data between DynamoDB and an Amazon EMR cluster. If you have large amounts of on-premise data to process, you may find the AWS Direct Connect service useful.

Topics

- [Upload Data to Amazon S3](#) (p. 45)
- [Import files with Distributed Cache](#) (p. 48)
- [How to Process Compressed Files](#) (p. 51)
- [Import DynamoDB Data into Hive](#) (p. 51)
- [Connect to Data with AWS DirectConnect](#) (p. 51)

- [Upload Large Amounts of Data with AWS Import/Export \(p. 51\)](#)

Upload Data to Amazon S3

For information on how to upload objects to Amazon S3, go to [Add an Object to Your Bucket](#) in the *Amazon Simple Storage Service Getting Started Guide*. For more information about using Amazon S3 with Hadoop, go to <http://wiki.apache.org/hadoop/AmazonS3>.

Topics

- [Create and Configure an Amazon S3 Bucket \(p. 45\)](#)
- [Best practices and recommended Amazon S3 Bucket Configuration \(p. 46\)](#)
- [Configure Multipart Upload for Amazon S3 \(p. 46\)](#)

Create and Configure an Amazon S3 Bucket

Amazon EMR uses the AWS SDK for Java with Amazon S3 to store input data, log files, and output data. Amazon S3 refers to these storage locations as *buckets*. Buckets have certain restrictions and limitations to conform with Amazon S3 and DNS requirements. For more information, go to [Bucket Restrictions and Limitations](#) in the *Amazon Simple Storage Service Developer Guide*.

This section shows you how to use the Amazon S3 AWS Management Console to create and then set permissions for an Amazon S3 bucket. However, you can also create and set permissions for an Amazon S3 bucket using the Amazon S3 API or the Curl command line tool. For information about Curl, go to [Amazon S3 Authentication Tool for Curl](#). For information about using the Amazon S3 API to create and configure an Amazon S3 bucket, see the [Amazon Simple Storage Service API Reference](#).

To create an Amazon S3 bucket using the console

1. Sign in to the AWS Management Console and open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.

2. Choose **Create Bucket**.

The **Create a Bucket** dialog box opens.

3. Enter a bucket name, such as `myawsbucket`.

This name should be globally unique, and cannot be the same name used by another bucket.

4. Select the **Region** for your bucket. To avoid paying cross-region bandwidth charges, create the Amazon S3 bucket in the same region as your cluster.

Refer to [Choose an AWS Region \(p. 25\)](#) for guidance on choosing a Region.

5. Choose **Create**.

You created a bucket with the URI `s3n://myawsbucket/`.

Note

If you enable logging in the **Create a Bucket** wizard, it enables only bucket access logs, not Amazon EMR cluster logs.

Note

For more information on specifying Region-specific buckets, see [Buckets and Regions](#) in the *Amazon Simple Storage Service Developer Guide* and [Available Region Endpoints for the AWS SDKs](#).

After you create your bucket you can set the appropriate permissions on it. Typically, you give yourself (the owner) read and write access and authenticated users read access.

To set permissions on an Amazon S3 bucket using the console

1. Sign in to the AWS Management Console and open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. In the **Buckets** pane, open (right-click) the bucket you just created.
3. Select **Properties**.
4. In the **Properties** pane, select the **Permissions** tab.
5. Choose **Add more permissions**.
6. Select **Authenticated Users** in the **Grantee** field.
7. To the right of the **Grantee** drop-down list, select **List**.
8. Choose **Save**.

You have created a bucket and restricted permissions to authenticated users.

Required Amazon S3 buckets must exist before you can create a cluster. You must upload any required scripts or data referenced in the cluster to Amazon S3. The following table describes example data, scripts, and log file locations.

Best practices and recommended Amazon S3 Bucket Configuration

The following are recommendations for using Amazon S3 Buckets with EMR clusters.

Enable Versioning

Versioning is a recommended configuration for your Amazon S3 bucket. By enabling versioning, you ensure that even if data is unintentionally deleted or overwritten it can be recovered. For more information, go to [Using Versioning](#) in the Amazon Simple Storage Service Developer Guide.

Lifecycle Management

Amazon S3 Lifecycle Management allows you to create rules which control the storage class and lifetime of your objects. EMR cluster components use multipart uploads via the AWS SDK for Java with Amazon S3 APIs to write log files and output data to Amazon S3. Amazon EMR does not automatically manage incomplete multipart uploads. Sometimes the upload of a large file can result in an incomplete Amazon S3 multipart upload. When a multipart upload is unable to complete successfully, the in-progress multipart upload continues to occupy your bucket and incurs storage charges. It is recommended that for buckets you use with Amazon EMR you enable a rule to remove incomplete multipart uploads three days after the upload initiation date.

When deleting an object in a versioned bucket a delete marker is created. If all previous versions of the object subsequently expire, an expired object delete marker is left within the bucket. While there is no charge for these delete markers, removing the expired delete markers can improve the performance of LIST requests. It is recommended that for versioned buckets you will use with Amazon EMR, you should also enable a rule to remove expired object delete markers. For more information, see [Lifecycle Configuration for a Bucket with Versioning](#) in the Amazon Simple Storage Service Console User Guide.

Performance best practices

Depending on your workloads, specific types of usage of EMR clusters and applications on those clusters can result in a high number of requests against your S3 bucket. For more information, go to [Request Rate and Performance Considerations](#) in the Amazon Simple Storage Service Developer Guide.

Configure Multipart Upload for Amazon S3

Important

You are responsible for the lifecycle management of your data stored in Amazon S3. For more information about lifecycle management, see [Best practices and recommended Amazon S3 Bucket Configuration](#) (p. 46).

Amazon EMR supports Amazon S3 multipart upload through the AWS SDK for Java. Multipart upload lets you upload a single object as a set of parts. You can upload these object parts independently and in any order. If transmission of any part fails, you can retransmit that part without affecting other parts. After all parts of your object are uploaded, Amazon S3 assembles the parts and creates the object.

For more information on Amazon S3 multipart uploads, go to [Uploading Objects Using Multipart Upload](#) in the *Amazon Simple Storage Service Developer Guide*.

The Amazon EMR configuration parameters for multipart upload are described in the following table.

Configuration Parameter Name	Default Value	Description
fs.s3n.multipart.uploads.enabled	True	A boolean type that indicates whether to enable multipart uploads.
fs.s3n.ssl.enabled	True	A boolean type that indicates whether to use http or https.
fs.s3.buckets.create.enabled	True	This setting automatically creates bucket if it doesn't exist. Setting to False will cause an exception on CreateBucket operations for that case.

You modify the configuration parameters for multipart uploads using a bootstrap action.

Disable Multipart Upload Using the Amazon EMR Console

This procedure explains how to disable multipart upload using the Amazon EMR console.

To disable multipart uploads with a bootstrap action using the console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**.
3. Choose **Go to advanced options**.
4. Choose **Edit Software Settings** and enter the following configuration: `classification=core-site,properties=[fs.s3n.multipart.uploads.enabled=false]`
5. Proceed with creating the cluster as described in [Plan and Configure Clusters \(p. 24\)](#).

Disable Multipart Upload Using the AWS CLI

This procedure explains how to disable multipart upload using the AWS CLI. To disable multipart upload, type the `create-cluster` command with the `--bootstrap-action` parameter.

To disable multipart upload using the AWS CLI

1. Create a file, `myConfig.json`, with the following contents:

```
[
  {
    "Classification": "core-site",
    "Properties": {
      "fs.s3n.multipart.uploads.enabled": "false"
    }
  }
]
```

```
] ]
```

2. Type the following command and replace `myKey` with the name of your EC2 key pair.

```
aws emr create-cluster --name "Test cluster" --release-label emr-4.0.0 --applications
Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --instance-
type m3.xlarge --instance-count 3 --configurations file:///myConfig.json
```

Disable Multipart Upload Using the API

For information on using Amazon S3 multipart uploads programmatically, go to [Using the AWS SDK for Java for Multipart Upload](#) in the *Amazon Simple Storage Service Developer Guide*.

For more information about the AWS SDK for Java, go to the [AWS SDK for Java](#) detail page.

Import files with Distributed Cache

Topics

- [Supported File Types](#) (p. 48)
- [Location of Cached Files](#) (p. 49)
- [Access Cached Files From Streaming Applications](#) (p. 49)
- [Access Cached Files From Streaming Applications Using the Amazon EMR Console](#) (p. 49)
- [Access Cached Files From Streaming Applications Using the AWS CLI](#) (p. 50)

Distributed Cache is a Hadoop feature that can boost efficiency when a map or a reduce task needs access to common data. If your cluster depends on existing applications or binaries that are not installed when the cluster is created, you can use Distributed Cache to import these files. This feature lets a cluster node read the imported files from its local file system, instead of retrieving the files from other cluster nodes.

For more information, go to <http://hadoop.apache.org/docs/stable/api/org/apache/hadoop/filecache/DistributedCache.html>.

You invoke Distributed Cache when you create the cluster. The files are cached just before starting the Hadoop job and the files remain cached for the duration of the job. You can cache files stored on any Hadoop-compatible file system, for example HDFS or Amazon S3. The default size of the file cache is 10GB. To change the size of the cache, reconfigure the Hadoop parameter, `local.cache.size` using the bootstrap action. For more information, see [\(Optional\) Create Bootstrap Actions to Install Additional Software](#) (p. 64).

Supported File Types

Distributed Cache allows both single files and archives. Individual files are cached as read only. Executables and binary files have execution permissions set.

Archives are one or more files packaged using a utility, such as `gzip`. Distributed Cache passes the compressed files to each slave node and decompresses the archive as part of caching. Distributed Cache supports the following compression formats:

- `zip`
- `tgz`
- `tar.gz`
- `tar`

- jar

Location of Cached Files

Distributed Cache copies files to slave nodes only. If there are no slave nodes in the cluster, Distributed Cache copies the files to the master node.

Distributed Cache associates the cache files to the current working directory of the mapper and reducer using symlinks. A symlink is an alias to a file location, not the actual file location. The value of the parameter, `yarn.nodemanager.local-dirs` in `yarn-site.xml`, specifies the location of temporary files. Amazon EMR sets this parameter to `/mnt/mapred`, or some variation based on instance type and EMR version. For example, a setting may have `/mnt/mapred` and `/mnt1/mapred` because the instance type has two ephemeral volumes. Cache files are located in a subdirectory of the temporary file location at `/mnt/mapred/taskTracker/archive`.

If you cache a single file, Distributed Cache puts the file in the `archive` directory. If you cache an archive, Distributed Cache decompresses the file, creates a subdirectory in `/archive` with the same name as the archive file name. The individual files are located in the new subdirectory.

You can use Distributed Cache only when using Streaming.

Access Cached Files From Streaming Applications

To access the cached files from your mapper or reducer applications, make sure that you have added the current working directory (`./`) into your application path and referenced the cached files as though they are present in the current working directory.

Access Cached Files From Streaming Applications Using the Amazon EMR Console

You can use the Amazon EMR console to create clusters that use Distributed Cache.

To specify Distributed Cache files using the console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**.
3. Choose **Step execution** as the Launch mode.
4. In the **Steps** section, in the **Add step** field, choose **Streaming program** from the list and click **Configure and add**.
5. In the **Arguments** field, include the files and archives to save to the cache and click **Add**.

The size of the file (or total size of the files in an archive file) must be less than the allocated cache size.

If you want to ...	Action	Example	
Add an individual file to the Distributed Cache	Specify <code>-cacheFile</code> followed by the name and location of the file, the pound (#) sign, and then the name you want to give the file when it's placed in the local cache.	<code>-cacheFile \s3://bucket_name/file_name#cache_file_name</code>	

If you want to ...	Action	Example	
Add an archive file to the Distributed Cache	Enter <code>-cacheArchive</code> followed by the location of the files in Amazon S3, the pound (#) sign, and then the name you want to give the collection of files in the local cache.	<code>-cacheArchive \ s3://bucket_name/ archive_name#cache_archive_name</code>	

- Proceed with configuring and launching your cluster. Your cluster copies the files to the cache location before processing any cluster steps.

Access Cached Files From Streaming Applications Using the AWS CLI

You can use the CLI to create clusters that use Distributed Cache.

To specify Distributed Cache files using the AWS CLI

- To submit a Streaming step when a cluster is created, type the `create-cluster` command with the `--steps` parameter. To specify Distributed Cache files using the AWS CLI, specify the appropriate arguments when submitting a Streaming step.

If you want to ...	Add the following parameter to the cluster ...
add an individual file to the Distributed Cache	specify <code>-cacheFile</code> followed by the name and location of the file, the pound (#) sign, and then the name you want to give the file when it's placed in the local cache.
add an archive file to the Distributed Cache	enter <code>-cacheArchive</code> followed by the location of the files in Amazon S3, the pound (#) sign, and then the name you want to give the collection of files in the local cache.

For more information on using Amazon EMR commands in the AWS CLI, see <http://docs.aws.amazon.com/cli/latest/reference/emr>.

Example 1

Type the following command to launch a cluster and submit a Streaming step that uses `-cacheFile` to add one file, `sample_dataset_cached.dat`, to the cache.

```
aws emr create-cluster --name "Test cluster" --release-label emr-4.0.0 --
applications Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey
--instance-type m3.xlarge --instance-count 3 --steps Type=STREAMING,Name="Streaming
program",ActionOnFailure=CONTINUE,Args=["--files","s3://my_bucket/my_mapper.py s3://
my_bucket/my_reducer.py","-mapper","my_mapper.py","-reducer","my_reducer.py","-input","s3://
my_bucket/my_input","-output","s3://my_bucket/my_output", "-cacheFile","s3://my_bucket/
sample_dataset.dat#sample_dataset_cached.dat"]
```

When you specify the instance count without using the `--instance-groups` parameter, a single Master node is launched, and the remaining instances are launched as core nodes. All nodes will use the instance type specified in the command.

If you have not previously created the default EMR service role and EC2 instance profile, type `aws emr create-default-roles` to create them before typing the `create-cluster` subcommand.

Example 2

The following command shows the creation of a streaming cluster and uses `-cacheArchive` to add an archive of files to the cache.

```
aws emr create-cluster --name "Test cluster" --release-label emr-4.0.0 --  
applications Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey  
--instance-type m3.xlarge --instance-count 3 --steps Type=STREAMING,Name="Streaming  
program",ActionOnFailure=CONTINUE,Args=["--files","s3://my_bucket/my_mapper.py s3://  
my_bucket/my_reducer.py","-mapper","my_mapper.py","-reducer","my_reducer.py","-input","s3://  
my_bucket/my_input","-output","s3://my_bucket/my_output", "-cacheArchive","s3://my_bucket/  
sample_dataset.tgz#sample_dataset_cached"]
```

When you specify the instance count without using the `--instance-groups` parameter, a single Master node is launched, and the remaining instances are launched as core nodes. All nodes will use the instance type specified in the command.

If you have not previously created the default EMR service role and EC2 instance profile, type `aws emr create-default-roles` to create them before typing the `create-cluster` subcommand.

How to Process Compressed Files

Hadoop checks the file extension to detect compressed files. The compression types supported by Hadoop are: gzip, bzip2, and LZO. You do not need to take any additional action to extract files using these types of compression; Hadoop handles it for you.

To index LZO files, you can use the `hadoop-lzo` library which can be downloaded from <https://github.com/kevinweil/hadoop-lzo>. Note that because this is a third-party library, Amazon EMR does not offer developer support on how to use this tool. For usage information, see [the hadoop-lzo readme file](#).

Import DynamoDB Data into Hive

The implementation of Hive provided by Amazon EMR includes functionality that you can use to import and export data between DynamoDB and an Amazon EMR cluster. This is useful if your input data is stored in DynamoDB. .

Connect to Data with AWS DirectConnect

AWS Direct Connect is a service you can use to establish a private dedicated network connection to AWS from your datacenter, office, or colocation environment. If you have large amounts of input data, using AWS Direct Connect may reduce your network costs, increase bandwidth throughput, and provide a more consistent network experience than Internet-based connections. For more information see the [AWS Direct Connect User Guide](#).

Upload Large Amounts of Data with AWS Import/Export

AWS Import/Export is a service you can use to transfer large amounts of data from physical storage devices into AWS. You mail your portable storage devices to AWS and AWS Import/Export transfers data directly off of your storage devices using Amazon's high-speed internal network. Your data load typically begins the next business day after your storage device arrives at AWS. After the data export or import completes, we return your storage device. For large data sets, AWS data transfer can be significantly faster than Internet transfer and more cost effective than upgrading your connectivity. For more information, see the [AWS Import/Export Developer Guide](#).

Configure an Output Location

The most common output format of an Amazon EMR cluster is as text files, either compressed or uncompressed. Typically, these are written to an Amazon S3 bucket. This bucket must be created before you launch the cluster. You specify the S3 bucket as the output location when you launch the cluster.

For more information, see the following topics:

Topics

- [Create and Configure an Amazon S3 Bucket \(p. 52\)](#)
- [What formats can Amazon EMR return? \(p. 53\)](#)
- [How to write data to an Amazon S3 bucket you don't own \(p. 53\)](#)
- [Compress the Output of your Cluster \(p. 55\)](#)

Create and Configure an Amazon S3 Bucket

Amazon EMR (Amazon EMR) uses Amazon S3 to store input data, log files, and output data. Amazon S3 refers to these storage locations as *buckets*. Buckets have certain restrictions and limitations to conform with Amazon S3 and DNS requirements. For more information, go to [Bucket Restrictions and Limitations](#) in the *Amazon Simple Storage Service Developers Guide*.

This section shows you how to use the Amazon S3 AWS Management Console to create and then set permissions for an Amazon S3 bucket. However, you can also create and set permissions for an Amazon S3 bucket using the Amazon S3 API or the third-party Curl command line tool. For information about Curl, go to [Amazon S3 Authentication Tool for Curl](#). For information about using the Amazon S3 API to create and configure an Amazon S3 bucket, go to the [Amazon Simple Storage Service API Reference](#).

To create an Amazon S3 bucket using the console

1. Sign in to the AWS Management Console and open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. Choose **Create Bucket**.

The **Create a Bucket** dialog box opens.

3. Enter a bucket name, such as `myawsbucket`.

This name should be globally unique, and cannot be the same name used by another bucket.

4. Select the **Region** for your bucket. To avoid paying cross-region bandwidth charges, create the Amazon S3 bucket in the same region as your cluster.

Refer to [Choose an AWS Region \(p. 25\)](#) for guidance on choosing a Region.

5. Choose **Create**.

You created a bucket with the URI `s3n://myawsbucket/`.

Note

If you enable logging in the **Create a Bucket** wizard, it enables only bucket access logs, not Amazon EMR cluster logs.

Note

For more information on specifying Region-specific buckets, refer to [Buckets and Regions](#) in the *Amazon Simple Storage Service Developer Guide* and [Available Region Endpoints for the AWS SDKs](#).

After you create your bucket you can set the appropriate permissions on it. Typically, you give yourself (the owner) read and write access and authenticated users read access.

To set permissions on an Amazon S3 bucket using the console

1. Sign in to the AWS Management Console and open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. In the **Buckets** pane, open (right-click) the bucket you just created.
3. Select **Properties**.
4. In the **Properties** pane, select the **Permissions** tab.
5. Choose **Add more permissions**.
6. Select **Authenticated Users** in the **Grantee** field.
7. To the right of the **Grantee** drop-down list, select **List**.
8. Choose **Save**.

You have created a bucket and restricted permissions to authenticated users.

Required Amazon S3 buckets must exist before you can create a cluster. You must upload any required scripts or data referenced in the cluster to Amazon S3. The following table describes example data, scripts, and log file locations.

Information	Example Location on Amazon S3
script or program	s3://myawsbucket/script/MapScript.py
log files	s3://myawsbucket/logs
input data	s3://myawsbucket/input
output data	s3://myawsbucket/output

What formats can Amazon EMR return?

The default output format for a cluster is text with key, value pairs written to individual lines of the text files. This is the output format most commonly used.

If your output data needs to be written in a format other than the default text files, you can use the Hadoop interface `OutputFormat` to specify other output types. You can even create a subclass of the `FileOutputFormat` class to handle custom data types. For more information, see <http://hadoop.apache.org/docs/current/api/org/apache/hadoop/mapred/OutputFormat.html>.

If you are launching a Hive cluster, you can use a serializer/deserializer (SerDe) to output data from HDFS to a given format. For more information, see <https://cwiki.apache.org/confluence/display/Hive/SerDe>.

How to write data to an Amazon S3 bucket you don't own

When you write a file to an Amazon Simple Storage Service (Amazon S3) bucket, by default, you are the only one able to read that file. The assumption is that you will write files to your own buckets, and this default setting protects the privacy of your files.

However, if you are running a cluster, and you want the output to write to the Amazon S3 bucket of another AWS user, and you want that other AWS user to be able to read that output, you must do two things:

- Have the other AWS user grant you write permissions for their Amazon S3 bucket. The cluster you launch runs under your AWS credentials, so any clusters you launch will also be able to write to that other AWS user's bucket.

- Set read permissions for the other AWS user on the files that you or the cluster write to the Amazon S3 bucket. The easiest way to set these read permissions is to use canned access control lists (ACLs), a set of pre-defined access policies defined by Amazon S3.

For information about how the other AWS user can grant you permissions to write files to the other user's Amazon S3 bucket, see [Editing Bucket Permissions](#) in the *Amazon Simple Storage Service Console User Guide*.

For your cluster to use canned ACLs when it writes files to Amazon S3, set the `fs.s3.canned.acl` cluster configuration option to the canned ACL to use. The following table lists the currently defined canned ACLs.

Canned ACL	Description
AuthenticatedRead	Specifies that the owner is granted <code>Permission.FullControl</code> and the <code>GroupGrantee.AuthenticatedUsers</code> group grantee is granted <code>Permission.Read</code> access.
BucketOwnerFullControl	Specifies that the owner of the bucket is granted <code>Permission.FullControl</code> . The owner of the bucket is not necessarily the same as the owner of the object.
BucketOwnerRead	Specifies that the owner of the bucket is granted <code>Permission.Read</code> . The owner of the bucket is not necessarily the same as the owner of the object.
LogDeliveryWrite	Specifies that the owner is granted <code>Permission.FullControl</code> and the <code>GroupGrantee.LogDelivery</code> group grantee is granted <code>Permission.Write</code> access, so that access logs can be delivered.
Private	Specifies that the owner is granted <code>Permission.FullControl</code> .
PublicRead	Specifies that the owner is granted <code>Permission.FullControl</code> and the <code>GroupGrantee.AllUsers</code> group grantee is granted <code>Permission.Read</code> access.
PublicReadWrite	Specifies that the owner is granted <code>Permission.FullControl</code> and the <code>GroupGrantee.AllUsers</code> group grantee is granted <code>Permission.Read</code> and <code>Permission.Write</code> access.

There are many ways to set the cluster configuration options, depending on the type of cluster you are running. The following procedures show how to set the option for common cases.

To write files using canned ACLs in Hive

- From the Hive command prompt, set the `fs.s3.canned.acl` configuration option to the canned ACL you want to have the cluster set on files it writes to Amazon S3. To access the Hive command prompt connect to the master node using SSH, and type Hive at the Hadoop command prompt. For more information, see [Connect to the Master Node Using SSH \(p. 173\)](#).

The following example sets the `fs.s3.canned.acl` configuration option to `BucketOwnerFullControl`, which gives the owner of the Amazon S3 bucket complete control over the file. Note that the set command is case sensitive and contains no quotation marks or spaces.

```
hive> set fs.s3.canned.acl=BucketOwnerFullControl;
```

```
hive> set fs.s3.canned.acl=BucketOwnerFullControl;
create table acl (n int) location 's3://acltestbucket/acl/';
insert overwrite table acl select count(n) from acl;
```

The last two lines of the example create a table that is stored in Amazon S3 and write data to the table.

To write files using canned ACLs in Pig

- From the Pig command prompt, set the `fs.s3.canned.acl` configuration option to the canned ACL you want to have the cluster set on files it writes to Amazon S3. To access the Pig command prompt connect to the master node using SSH, and type Pig at the Hadoop command prompt. For more information, see [Connect to the Master Node Using SSH \(p. 173\)](#).

The following example sets the `fs.s3.canned.acl` configuration option to `BucketOwnerFullControl`, which gives the owner of the Amazon S3 bucket complete control over the file. Note that the set command includes one space before the canned ACL name and contains no quotation marks.

```
pig> set fs.s3.canned.acl BucketOwnerFullControl;
store some data into 's3://acltestbucket/pig/acl';
```

To write files using canned ACLs in a custom JAR

- Set the `fs.s3.canned.acl` configuration option using Hadoop with the `-D` flag. This is shown in the example below.

```
hadoop jar hadoop-examples.jar wordcount
-Dfs.s3.canned.acl=BucketOwnerFullControl s3://mybucket/input s3://mybucket/output
```

Compress the Output of your Cluster

Topics

- [Output Data Compression \(p. 55\)](#)
- [Intermediate Data Compression \(p. 56\)](#)
- [Using the Snappy Library with Amazon EMR \(p. 56\)](#)

Output Data Compression

This compresses the output of your Hadoop job. If you are using `TextOutputFormat` the result is a gzipped text file. If you are writing to `SequenceFiles` then the result is a `SequenceFile` which is compressed internally. This can be enabled by setting the configuration setting `mapred.output.compress` to `true`.

If you are running a streaming job you can enable this by passing the streaming job these arguments.

```
-jobconf mapred.output.compress=true
```

You can also use a bootstrap action to automatically compress all job outputs. Here is how to do that with the Ruby client.

```
--bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop \  
--args "-s,mapred.output.compress=true"
```

Finally, if are writing a Custom Jar you can enable output compression with the following line when creating your job.

```
FileOutputFormat.setCompressOutput(conf, true);
```

Intermediate Data Compression

If your job shuffles a significant amount data from the mappers to the reducers, you can see a performance improvement by enabling intermediate compression. Compresses the map output and decompresses it when it arrives on the slave node. The configuration setting is `mapred.compress.map.output`. You can enable this similarly to output compression.

When writing a Custom Jar, use the following command:

```
conf.setCompressMapOutput(true);
```

Using the Snappy Library with Amazon EMR

Snappy is a compression and decompression library that is optimized for speed. It is available on Amazon EMR AMIs version 2.0 and later and is used as the default for intermediate compression. For more information about Snappy, go to <http://code.google.com/p/snappy/>.

Configure a Cluster to be Transient or Long-Running

You can run your cluster as a transient process: one that launches the cluster, loads the input data, processes the data, stores the output results, and then automatically shuts down. This is the standard model for a cluster that performs a periodic processing task. Shutting down the cluster automatically ensures that you are only billed for the time required to process your data.

The other model for running a cluster is as a long-running cluster. In this model, the cluster launches and loads the input data. From there, you might interactively query the data, use the cluster as a data warehouse, or do periodic processing on a data set so large that it would be inefficient to load the data into new clusters each time. In this model, the cluster persists even when there are no tasks queued for processing. Another option you might want to enable on a long-running cluster is termination protection. This protects your cluster from being terminated accidentally or in the event that an error occurs. For more information, see [Managing Cluster Termination \(p. 188\)](#).

By default, clusters you create are long-running clusters. If you use quick create in the console or don't specify an option when using `create-cluster` from the command line or the API, `auto-terminate` is disabled. However, clusters launched using the API have `auto-terminate` enabled. You can specify auto-termination using the console, the AWS CLI, or programmatically using the [KeepJobFlowAliveWhenNoSteps](#) parameter while executing the [RunJobFlow](#) action.

To launch a transient cluster using the console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**.
3. Choose **Go to advanced options**.
4. Under **Add steps (optional)** select **Auto-terminate cluster after the last step is completed**.
5. Proceed with creating the cluster as described in [Plan and Configure Clusters \(p. 24\)](#).

To launch a transient cluster using the AWS CLI

Specify the `--auto-terminate` parameter when you use the `create-cluster` command to create a transient cluster.

- The following example demonstrates using the `--auto-terminate` parameter. You can type the following command and replace `myKey` with the name of your EC2 key pair.

Note

Linux line continuation characters (`\`) are included for readability. They can be removed or used in Linux commands. For Windows, remove them or replace with a caret (^).

```
aws emr create-cluster --name "Test cluster" --release-label emr-4.1.0emr-4.2.0 \
--applications Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey \
--instance-type m3.xlarge --instance-count 3 --auto-terminate
```

Note

If you have not previously created the default EMR service role and EC2 instance profile, type `aws emr create-default-roles` to create them before you use the `create-cluster` command.

For more information on using Amazon EMR commands in the AWS CLI, see [AWS CLI Reference](#).

Using a Custom AMI

Amazon EMR uses an Amazon Linux Amazon Machine Image (AMI) to initialize Amazon EC2 instances when you create and launch a cluster. The AMI contains the Amazon Linux operating system, other software, and configurations required for each instance to host your cluster applications. By default, when you create a cluster, you don't need to think about the AMI. When EC2 instances in your cluster launch, Amazon EMR starts with a default Amazon Linux AMI that Amazon EMR owns, runs any bootstrap actions you specify, and then installs and configures the applications and components that you select. You can also customize the Amazon EBS root device volume size of the default AMI. For more information, see [the section called "Specifying the Amazon EBS Root Device Volume Size" \(p. 62\)](#).

Alternatively, if you use Amazon EMR version 5.7.0 or later, you can specify a custom Amazon Linux AMI when you create a cluster and customize its root volume size as well. When each EC2 instance launches, it starts with your custom AMI instead of the EMR-owned AMI.

Specifying a custom AMI is useful for the following use cases:

- Encrypt the EBS root device volumes (boot volumes) of EC2 instances in your cluster. For more information, see [Creating a Custom AMI with an Encrypted Amazon EBS Root Device Volume \(p. 61\)](#).

Note

A custom AMI is only required when you need to encrypt the EBS root device volumes of instances. You can also encrypt EBS storage volumes using Amazon EMR security

configuration to enable at-rest encryption. For more information, see [At-Rest Encryption for Local Disks](#) in the *Amazon EMR Release Guide*.

- Pre-install applications and perform other customizations instead of using bootstrap actions, which can improve cluster start time and streamline the startup work flow. For more information and an example, see [Creating a Custom Amazon Linux AMI from a Preconfigured Instance](#) (p. 59).
- Implement more sophisticated cluster and node configurations than bootstrap actions allow.

For more information, see [Amazon Machine Images \(AMI\)](#). For more information about using bootstrap actions, see [\(Optional\) Create Bootstrap Actions to Install Additional Software](#) (p. 64).

Best Practices and Considerations

When you create a custom AMI for Amazon EMR, consider the following:

- You must use an Amazon Linux AMI.
- Only 64-bit Amazon Linux AMIs are supported.
- Base your customization on the most recent EBS-backed [Amazon Linux AMI](#).
- Do not copy a snapshot of an existing Amazon EMR instance to create a custom AMI. This causes errors.
- Only the HVM virtualization type and instances compatible with Amazon EMR are supported. Be sure to select the HVM image and an instance type compatible with Amazon EMR as you go through the AMI customization process. For compatible instances and virtualization types, see [Supported Instance Types](#) (p. 69).
- Your service role must have launch permissions on the AMI, so either the AMI must be public, or you must be the owner of the AMI or have it shared with you by the owner.
- Creating users on the AMI with the same name as applications causes errors (for example, `hadoop`, `hdfs`, `yarn`, or `spark`).
- The contents of `/tmp`, `/var`, and `/emr`—if they exist on the AMI—are moved to `/mnt/tmp`, `/mnt/var`, and `/mnt/emr` respectively during startup. Files are preserved, but if there is a large amount of data, startup may take longer than expected.

For more information, see [Creating an Amazon EBS-Backed Linux AMI](#) in the *Amazon EC2 User Guide for Linux Instances*.

Specifying a Custom AMI

You can specify a custom AMI ID when you create a cluster using the AWS Management Console, AWS CLI, or the Amazon EMR API. The AMI must exist in the same region where you create the cluster.

To specify a custom AMI using the console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**, **Go to advanced options**.
3. Under **Software Configuration**, for **Release**, choose **emr-5.7.0** or later and then choose other options as appropriate for your application. Choose **Next**.
4. Select values under **Hardware Configuration** that are appropriate for your application, and choose **Next**.
5. Under **Additional Options**, for **Custom AMI ID**, enter a value and leave the update option selected. For more information about changing the update option, see [Managing AMI Package Repository Updates](#) (p. 59).

Tags ⓘ

Key	Value (optional)
<input type="text" value="Add a key to create a tag"/>	

Additional Options

☐ EMRFS consistent view ⓘ

Custom AMI ID ⓘ

☒ Update all installed packages on reboot (recommended)

▶ Bootstrap Actions

6. To launch the cluster, choose **Next** and complete other configuration options.

To specify a custom AMI using the AWS CLI

- Use the `--custom-ami-id` parameter to specify the AMI ID when you run the `aws emr create-cluster` command.

The following example specifies a cluster that uses a custom AMI with a 20 GiB boot volume. For more information, see [Specifying the Amazon EBS Root Device Volume Size \(p. 62\)](#).

```
aws emr create-cluster --name "Cluster with My Custom AMI" \  
    --custom-ami-id ami-5d1f626 --ebs-root-volume-size 20 \  
    --release-label emr-5.7.0 --use-default-roles \  
    --instance-count 2 --instance-type m3.xlarge
```

Managing AMI Package Repository Updates

On first boot, by default, Amazon Linux AMIs connect to package repositories to install security updates before other services start. Depending on your requirements, you may choose to disable these updates when you specify a custom AMI for Amazon EMR. The option to disable this feature is available only when you use a custom AMI.

Warning

We strongly recommend that you choose to update all installed packages on reboot when you specify a custom AMI. Choosing not to update packages creates additional security risks.

Using the AWS Management Console, you can select the option to disable updates when you choose **Custom AMI ID**.

Using the AWS CLI, you can specify `--repo-upgrade-on-boot NONE` along with `--custom-ami-id` when using the `create-cluster` command.

Using the Amazon EMR API, you can specify `NONE` for the `RepoUpgradeOnBoot` parameter.

Creating a Custom Amazon Linux AMI from a Preconfigured Instance

The basic steps for pre-installing software and performing other configurations to create a custom Amazon Linux AMI for Amazon EMR are as follows:

- Launch an instance from the base Amazon Linux AMI.

- Connect to the instance to install software and perform other customizations.
- Create a new image (AMI snapshot) of the instance you configured.

After you create the image based on your customized instance, you can copy that image to an encrypted target as described in [Creating a Custom AMI with an Encrypted Amazon EBS Root Device Volume](#) (p. 61).

Tutorial: Creating an AMI from an Instance with Custom Software Installed

To launch an EC2 instance based on the most recent Amazon Linux AMI

1. Use the AWS CLI to run the following command, which creates an instance from an existing AMI. Replace *MyKeyName* with the key pair you use to connect to the instance. The value *ami-6df1e514* specifies the most recent Amazon Linux AMI available at the time of writing (Amazon EMR version 5.7.0).

```
aws ec2 run-instances --image-id ami-6df1e514 --count 1 --instance-type m3.xlarge --  
key-name MyKeyName --region us-west-2
```

The *InstanceId* output value is used as *MyInstanceId* in the next step.

2. Run the following command:

```
aws ec2 describe-instances --instance-ids MyInstanceId
```

The *PublicDnsName* output value is used to connect to the instance in the next step.

To connect to the instance and install software

1. Use an SSH connection that lets you run shell commands on your Linux instance. For more information, see [Connecting to Your Linux Instance Using SSH](#) in the *Amazon EC2 User Guide for Linux Instances*.
2. Perform any required customizations. For example:

```
sudo yum install MySoftwarePackage  
sudo pip install MySoftwarePackage
```

To create a snapshot from your customized image

- After you customize the instance, use the `create-image` command to create an AMI from the instance.

```
aws ec2 create-image --no-dry-run --instance-id MyInstanceId --name MyEmrCustomAmi
```

The *imageID* output value is used when you launch the cluster or create an encrypted snapshot. For more information, see [Specifying a Custom AMI](#) (p. 58) and [Creating a Custom AMI with an Encrypted Amazon EBS Root Device Volume](#) (p. 61).

Creating a Custom AMI with an Encrypted Amazon EBS Root Device Volume

To encrypt the Amazon EBS root device volume of an Amazon Linux AMI for Amazon EMR, copy a snapshot image from an unencrypted AMI to an encrypted target. For information about creating encrypted EBS volumes, see [Amazon EBS encryption](#) in the *Amazon EC2 User Guide for Linux Instances*. The source AMI for the snapshot can be the base Amazon Linux AMI, or you can copy a snapshot from an AMI derived from the base Amazon Linux AMI that you customized.

This encryption method only applies to the EBS root device volumes. Use the local-disk encryption feature of security configurations (Amazon EMR version 4.8 or later), to encrypt EBS storage volumes. For more information, see [At-Rest Encryption for Local Disks](#).

You can use an external key provider or an AWS customer master key (CMK) to encrypt the EBS root volume. The service role that Amazon EMR uses (usually the default `EMR_DefaultRole`) must be allowed to encrypt and decrypt the volume, at minimum, for Amazon EMR to create a cluster using the AMI. When using AWS KMS as the key provider, this means the `kms:encrypt` and `kms:decrypt` actions must be allowed. The easiest way to do this is to add the role as a key user as described in the following tutorial.

Note

Local disk encryption using a security configuration requires that the Amazon EMR *instance role* (usually `ERM_EC2_DefaultRole`) have permission rather than the Amazon EMR *service role* (usually `EMR_DefaultRole`). You can use the same key for both purposes, but both the service role and the instance role must be attached as key policy users.

Tutorial: Creating a Custom AMI with an Encrypted Root Device Volume Using a KMS CMK

The first step in this example is to find the ARN of a KMS CMK or create a new one. For more information about creating keys, see [Creating Keys](#) in the *AWS Key Management Service Developer Guide*. The following procedure shows you how to add the default service role, `EMR_DefaultRole`, as a key user to the key policy. Write down the **ARN** value for the key as you create or edit it. You use the ARN later, when you create the AMI.

To add the service role for Amazon EC2 to the list of encryption key users using the console

1. Open the **Encryption Keys** section of the Identity and Access Management (IAM) console at <https://console.aws.amazon.com/iam/home#encryptionKeys>.
2. For **Region**, choose the appropriate AWS region. Do not use the region selector in the navigation bar (top right corner).
3. Choose the alias of the CMK to use.
4. On the key details page under **Key Users**, choose **Add**.
5. In the **Attach** dialog box, choose the Amazon EMR service role. The name of the default role is `EMR_DefaultRole`.
6. Choose **Attach**.

To create an encrypted AMI using the AWS CLI

- Use the `aws ec2 copy-image` command from the AWS CLI to create an AMI with an encrypted EBS root device volume and the key that you modified. Replace the `--kms-key-id` value specified with the full ARN of the key that you created or modified earlier.

```
aws ec2 copy-image --source-image-id ami-6df1e514 --source-region us-west-2 --  
name MyEncryptedEMRAmi --encrypted --kms-key-id arn:aws:kms:us-west-1:210883312745:key/  
c1e55ccd-6628-4937-adad-925be8e39991
```

The output of the command provides the ID of the AMI that you created, which you can specify when you create a cluster. For more information, see [Specifying a Custom AMI \(p. 58\)](#). You can also choose to customize this AMI by installing software and performing other configurations. For more information, see [Creating a Custom Amazon Linux AMI from a Preconfigured Instance \(p. 59\)](#).

Specifying the Amazon EBS Root Device Volume Size

Whether you use the default AMI or a custom AMI, you can specify the size of the EBS root device volume in GiB. This option is available only with Amazon EMR version 4.x and later. You can specify the volume size from 10 GiB (the default) up to 100 GiB when you create a cluster using the AWS Management Console, the AWS CLI, or the Amazon EMR API. This sizing applies only to the EBS root device volume and applies to all instances in the cluster. It does not apply to storage volumes, which you specify separately for each instance type when you create your cluster.

Note

If you use the default AMI, Amazon EMR attaches General Purpose SSD (gp2) as the root device volume type. A custom AMI may have a different root device volume type. For more information, see [Specifying a Custom AMI \(p. 58\)](#).

The cost of the EBS root device volume is pro-rated by the hour, based on the monthly EBS charges for that volume type in the region where the cluster runs. The same is true of storage volumes. Charges are in GB, but you specify the size of the root volume in GiB, so you may want to consider this in your estimates (1 GB is 0.931323 GiB). To estimate the charges associated with EBS root device volumes in your cluster, use the following formula:

$$(\$EBS\text{ GBmonth}) \times 0.931323 \div 30 \div 24 \times EMR_EBSRootGiB \times InstanceCount$$

For example, take a cluster that has a master node, a core node, and uses the base Amazon Linux AMI, with the default 10 GiB root device volume. If the EBS cost in the region is USD\$0.10/GB-Month, that works out to be approximately \$0.00129 per instance per hour and \$0.00258 per hour for the cluster (\$0.10 GB-month divided by 30 days, divided by 24 hours, multiplied by 10 GB, multiplied by 2 cluster instances).

To specify the EBS root device volume size using the console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**.
3. Choose **Go to advanced options**.
4. Under **Software Configuration**, for **Release**, choose a 4.x or 5.x value and other options as appropriate for your application, and choose **Next**.
5. Under **Hardware Configuration**, for **Root device EBS volume size**, enter a value between 10 GiB and 100 GiB.

Services ▾ Resource Groups ▾ ☆

Create Cluster - Advanced Options [Go to quick options](#)

Step 1: Software and Steps
Step 2: Hardware
Step 3: General Cluster Settings
Step 4: Security

Hardware Configuration ⓘ

If you need more than 20 EC2 instances, [see this topic](#).

Instance group configuration ☒ **Uniform instance groups**
Specify a single instance type and purchasing option for each node type.

☐ **Instance fleets**
Specify target capacity and how Amazon EMR fulfills it for each node type. Mix instance types and purchasing options. [Learn more](#)

Network [Create a VPC ⓘ](#)

EC2 availability zone

Root device EBS volume size GiB ⓘ

To specify the EBS root device volume size using the AWS CLI

- Use the `--ebs-root-volume-size` parameter of the `create-cluster` command.

```
aws emr create-cluster --release-label emr-5.7.0 --ebs-root-volume-size 20 --  
instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m3.xlarge  
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m3.xlarge
```

Configure Cluster Software

When you select a software release, Amazon EMR uses an Amazon Machine Image (AMI) with Amazon Linux to install the software that you choose when you launch your cluster, such as Hadoop, Spark, and Hive. Amazon EMR provides new releases on a regular basis, adding new features, new applications, and general updates. We recommend that you use the latest release to launch your cluster whenever possible. The latest release is the default option when you launch a cluster from the console.

For more information about Amazon EMR releases and versions of software available with each release, go to the [Amazon EMR Release Guide](#). For more information about how to edit the default configurations of applications and software installed on your cluster, go to [Configuring Applications](#) in the Amazon EMR Release Guide. Some versions of the open-source Hadoop and Spark ecosystem components that are included in Amazon EMR releases have patches and improvements, which are documented in the [Amazon EMR Release Guide](#).

In addition to the standard software and applications that are available for installation on your cluster, you can use bootstrap actions to install custom software. Bootstrap actions are scripts that run on the instances when your cluster is launched, and that run on new nodes that are added to your cluster when they are created. Bootstrap actions are also useful to invoke AWS CLI commands on each node to copy objects from Amazon S3 to each node in your cluster.

Note

Bootstrap actions are used differently in Amazon EMR release 4.x and later. For more information about these differences from Amazon EMR AMI versions 2.x and 3.x, go to [Differences Introduced in 4.x](#) in the Amazon EMR Release Guide.

(Optional) Create Bootstrap Actions to Install Additional Software

You can use a *bootstrap action* to install additional software on your cluster. Bootstrap actions are scripts that are run on the cluster nodes when Amazon EMR launches the cluster. They run before Amazon EMR installs specified applications and the node begins processing data. If you add nodes to a running cluster, bootstrap actions run on those nodes also. You can create custom bootstrap actions and specify them when you create your cluster.

Note

Most predefined bootstrap actions for Amazon EMR AMI versions 2.x and 3.x are not supported in Amazon EMR releases 4.x. For example, `configure-Hadoop` and `configure-daemons` are not supported in Amazon EMR release 4.x. Instead, Amazon EMR release 4.x natively provides this functionality. For more information about how to migrate bootstrap actions from Amazon EMR AMI versions 2.x and 3.x to Amazon EMR release 4.x, go to [Differences Introduced in 4.x](#) in the Amazon EMR Release Guide.

Contents

- [Bootstrap Action Basics](#) (p. 64)
- [Run If Bootstrap Action](#) (p. 64)
- [Shutdown Actions](#) (p. 65)
- [Use Custom Bootstrap Actions](#) (p. 65)

Bootstrap Action Basics

Bootstrap actions execute as the Hadoop user by default. You can execute a bootstrap action with root privileges by using `sudo`.

All Amazon EMR management interfaces support bootstrap actions. You can specify up to 16 bootstrap actions per cluster by providing multiple `bootstrap-action` parameters from the console, AWS CLI, or API.

From the Amazon EMR console, you can optionally specify a bootstrap action while creating a cluster.

When you use the CLI, you can pass references to bootstrap action scripts to Amazon EMR by adding the `--bootstrap-action` parameter when you create the cluster using the `create-cluster` command. The syntax for a `--bootstrap-action` parameter is as follows:

AWS CLI

```
--bootstrap-action Path=s3://mybucket/filename", Args=[arg1,arg2]
```

If the bootstrap action returns a nonzero error code, Amazon EMR treats it as a failure and terminates the instance. If too many instances fail their bootstrap actions, then Amazon EMR terminates the cluster. If just a few instances fail, Amazon EMR attempts to reallocate the failed instances and continue. Use the cluster `lastStateChangeReason` error code to identify failures caused by a bootstrap action.

Run If Bootstrap Action

Amazon EMR provides this predefined bootstrap action to run a command conditionally when an instance-specific value is found in the `instance.json` or `job-flow.json` file. The command can refer to a file in Amazon S3 that Amazon EMR can download and execute.

The location of the script is `s3://elasticmapreduce/bootstrap-actions/run-if`.

The following example echoes the string "running on master node" if the node is a master.

To run a command conditionally using the AWS CLI

When using the AWS CLI to include a bootstrap action, specify the `Path` and `Args` as a comma-separated list.

- To launch a cluster with a bootstrap action that conditionally runs a command when an instance-specific value is found in the `instance.json` or `job-flow.json` file, type the following command and replace `myKey` with the name of your EC2 key pair.

```
aws emr create-cluster --name "Test cluster" --release-label emr-4.0.0 --use-default-roles --ec2-attributes KeyName=myKey --applications Name=Hive --instance-count 1 --instance-type m3.xlarge --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/run-if,Args=["instance.isMaster=true", "echo running on master node"]
```

When you specify the instance count without using the `--instance-groups` parameter, a single Master node is launched, and the remaining instances are launched as core nodes. All nodes will use the instance type specified in the command.

Note

If you have not previously created the default Amazon EMR service role and EC2 instance profile, type `aws emr create-default-roles` to create them before typing the `create-cluster` subcommand.

For more information on using Amazon EMR commands in the AWS CLI, see <http://docs.aws.amazon.com/cli/latest/reference/emr>.

Shutdown Actions

A bootstrap action script can create one or more shutdown actions by writing scripts to the `/mnt/var/lib/instance-controller/public/shutdown-actions/` directory. When a cluster is terminated, all the scripts in this directory are executed in parallel. Each script must run and complete within 60 seconds.

Shutdown action scripts are not guaranteed to run if the node terminates with an error.

Note

When using Amazon EMR versions 4.0 and later, you must manually create the `/mnt/var/lib/instance-controller/public/shutdown-actions/` directory on the master node. It doesn't exist by default; however, after being created, scripts in this directory nevertheless run before shutdown. For more information about connecting to the Master node to create directories, see [Connect to the Master Node Using SSH \(p. 173\)](#).

Use Custom Bootstrap Actions

You can create a custom script to perform a customized bootstrap action. Any of the Amazon EMR interfaces can reference a custom bootstrap action.

Contents

- [Add Custom Bootstrap Actions Using the AWS CLI or the Amazon EMR CLI \(p. 65\)](#)
- [Add Custom Bootstrap Actions Using the Console \(p. 66\)](#)
- [Use a Custom Bootstrap Action to Copy an Object from Amazon S3 to Each Node \(p. 67\)](#)

Add Custom Bootstrap Actions Using the AWS CLI or the Amazon EMR CLI

The following example uses a bootstrap action script to download and extracts a compressed TAR archive from Amazon S3. The sample script is stored at <http://elasticmapreduce.s3.amazonaws.com/bootstrap-actions/download.sh>.

The sample script looks like the following:

```
#!/bin/bash
set -e
wget -S -T 10 -t 5 http://elasticmapreduce.s3.amazonaws.com/bootstrap-actions/file.tar.gz
mkdir -p /home/hadoop/contents
tar -xzf file.tar.gz -C /home/hadoop/contents
```

To create a cluster with a custom bootstrap action using the AWS CLI

When using the AWS CLI to include a bootstrap action, specify the `Path` and `Args` as a comma-separated list. The following example does not use an arguments list.

- To launch a cluster with a custom bootstrap action, type the following command, replace `myKey` with the name of your EC2 key pair.

- Linux, UNIX, and Mac OS X users:

```
aws emr create-cluster --name "Test cluster" --release-label emr-4.0.0 \
--use-default-roles --ec2-attributes KeyName=myKey \
--applications Name=Hive Name=Pig \
--instance-count 3 --instance-type m3.xlarge \
--bootstrap-action Path="s3://elasticmapreduce/bootstrap-actions/download.sh"
```

- Windows users:

```
aws emr create-cluster --name "Test cluster" --release-label emr-4.2.0 --use-default-
roles --ec2-attributes KeyName=myKey --applications Name=Hive Name=Pig --instance-
count 3 --instance-type m3.xlarge --bootstrap-action Path="s3://elasticmapreduce/
bootstrap-actions/download.sh"
```

When you specify the instance count without using the `--instance-groups` parameter, a single Master node is launched, and the remaining instances are launched as core nodes. All nodes will use the instance type specified in the command.

Note

If you have not previously created the default Amazon EMR service role and EC2 instance profile, type `aws emr create-default-roles` to create them before typing the `create-cluster` subcommand.

For more information on using Amazon EMR commands in the AWS CLI, see <http://docs.aws.amazon.com/cli/latest/reference/emr>.

Add Custom Bootstrap Actions Using the Console

The following procedure describes how to use your own custom bootstrap action.

To create a cluster with a custom bootstrap action using the console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**.
3. Click **Go to advanced options**.
4. In Create Cluster - Advanced Options, Steps 1 and 2 choose the options as desired and proceed to **Step 3: General Cluster Settings**.
5. Under **Bootstrap Actions** select **Configure and add** to specify the Name, JAR location, and arguments for your bootstrap action. Choose **Add**.

6. Optionally add more bootstrap actions as desired.
7. Proceed to create the cluster. Your bootstrap action(s) will be performed after the cluster has been provisioned and initialized.

While the cluster's master node is running, you can connect to the master node and see the log files that the bootstrap action script generated in the `/mnt/var/log/bootstrap-actions/1` directory.

Related Topics

- [View Log Files \(p. 146\)](#)

Use a Custom Bootstrap Action to Copy an Object from Amazon S3 to Each Node

You can use a bootstrap action to copy objects from Amazon S3 to each node in your Amazon EMR cluster before your applications are installed. The AWS CLI is installed on each node of your cluster, and your bootstrap action can call these commands. The following is an example bootstrap action script that copies an object from Amazon S3 to a local folder on your cluster:

```
aws s3 cp s3://mybucket/myfolder/myfile /mnt1/myfolder
```

Configure Cluster Hardware and Networking

An important consideration when you create an EMR cluster is how you configure Amazon EC2 instances and network options. EC2 instances in an EMR cluster are organized into *node types*. There are three node types in an Amazon EMR cluster: the *master node*, the *core node*, and *task nodes*. Each node performs a set of roles defined by the distributed applications that you install on the cluster. During a Hadoop MapReduce or Spark job, for example, components on core and task nodes process the data, transfer the output to Amazon S3 or HDFS, and provide status metadata back to the master node. In the case of a single node cluster, all components run on the master node.

The collection of EC2 instances that host each node type is called either an *instance fleet* or a *uniform instance group*. The instance fleets or uniform instance groups configuration is a choice you make when you create a cluster. It applies to all node types, and it can't be changed later.

Note

The instance fleets configuration is available only in Amazon EMR versions 4.8.0 and later, excluding 5.0.x versions.

Master Node

The master node manages the cluster and typically runs master components of distributed applications. For example, the master node runs the YARN ResourceManager service to manage resources for applications, as well as the HDFS NameNode service. It also tracks the status of jobs submitted to the cluster and monitors the health of the instance groups. To monitor the progress of a cluster, you can SSH into the master node as the Hadoop user and either look at the Hadoop log files directly or access the user interface that Hadoop or Spark publishes to a web server running on the master node. For more information, see [View Log Files \(p. 146\)](#). Because there is only one master node, the instance group or instance fleet consists of a single EC2 instance.

Core Nodes

Core nodes are managed by the master node. Core nodes run the Data Node daemon to coordinate data storage as part of the Hadoop Distributed File System (HDFS). They also run the Task Tracker daemon

and perform other parallel computation tasks on data that installed applications require. For example, a core node runs YARN NodeManager daemons, Hadoop MapReduce tasks, and Spark executors. Like the master node, at least one core node is required per cluster. However, unlike the master node, there can be multiple core nodes—and therefore multiple EC2 instances—in the instance group or instance fleet. There is only one core instance group or instance fleet. With instance groups, you can add and remove EC2 instances while the cluster is running or set up automatic scaling. For more information about adding and removing EC2 instances with the instance groups configuration, see [Scaling Cluster Resources \(p. 190\)](#). With instance fleets, you can effectively add and remove instances by modifying the instance fleet's target capacities for On-Demand and Spot accordingly. For more information about target capacities, see [Instance Fleet Options \(p. 86\)](#).

Warning

Removing HDFS daemons from a running node runs the risk of losing data.

Task Nodes

Task nodes are optional. You can use them to add power to perform parallel computation tasks on data, such as Hadoop MapReduce tasks and Spark executors. Task nodes don't run the Data Node daemon, nor do they store data in HDFS. As with core nodes, you can add task nodes to a cluster by adding EC2 instances to an existing uniform instance group, or modifying target capacities for a task instance fleet. Clusters with the uniform instance group configuration can have up to a total of 48 task instance groups. The ability to add uniform instance groups in this way allows you to mix EC2 instance types and pricing options, such as On-Demand Instances and Spot Instances. This gives you flexibility to respond to workload requirements in a cost-effective way. When you use the instance fleet configuration for your cluster, the ability to mix instance types and purchasing options is built in, so there is only one task instance fleet.

Instance Fleets

The instance fleets configuration offers the widest variety of provisioning options for EC2 instances. Each node type has a single instance fleet, and the task instance fleet is optional. For each instance fleet, you specify up to five instance types, which can be provisioned as On-Demand and Spot Instances. For the core and task instance fleets, you assign a *target capacity* for On-Demand Instances, and another for Spot Instances. Amazon EMR chooses any mix of the five instance types to fulfill the target capacities, provisioning both On-Demand and Spot Instances. For the master node type, Amazon EMR chooses a single instance type from your list of up to five, and you specify whether it's provisioned as an On-Demand or Spot Instance. Instance fleets also provide additional options for Spot Instance purchases, which include a defined duration (also known as a spot block) and a timeout that specifies an action to take if Spot capacity can't be provisioned. For more information, see [Configure Instance Fleets \(p. 86\)](#).

Uniform Instance Groups

Uniform instance groups offer a simplified setup. Each Amazon EMR cluster can include up to 50 instance groups: one master instance group that contains one EC2 instance, a core instance group that contains one or more EC2 instances, and up to 48 optional task instance groups. Each core and task instance group can contain any number of EC2 instances. You can scale each instance group by adding and removing EC2 instances manually, or you can set up automatic scaling. For more information about configuring uniform instance groups, see [Configure Uniform Instance Groups \(p. 93\)](#). For information about adding and removing instances, see [Scaling Cluster Resources \(p. 190\)](#).

Plan and Configure EC2 Instances

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 the instance type, you can choose different purchasing options for EC2 instances. You can specify different instance types and purchasing

options within instance groups or instance fleets. For more information about instance fleets, see [Configure Instance Fleets](#) (p. 86). For more information about instance groups, see [Create a Cluster with Instance Fleets or Uniform Instance Groups](#) (p. 85). For guidance about choosing the right options, see [Cluster Configuration Guidelines](#) (p. 95).

Topics

- [Supported Instance Types](#) (p. 69)
- [Instance Purchasing Options](#) (p. 71)
- [Instance Store and Amazon EBS](#) (p. 73)

Supported Instance Types

Amazon EC2 offers several basic instance types.

- **General purpose**—You can use Amazon EC2 general purposes instances for most applications.
- **Compute optimized**—These instances have proportionally more CPU resources than memory (RAM) for compute-intensive applications. Cluster compute instances also have increased network performance.
- **Memory optimized**—These instances offer large memory sizes for high throughput applications, including database and memory caching applications.
- **Storage optimized**—These instances provide proportionally high storage resources. They are well suited for data warehouse applications.
- **GPU instances**—These instances provide compute resources for increased parallel processing performance with GPU-assisted applications.

The following table describes the instance types that Amazon EMR supports. For more information on these instance types, families, and virtualization types, see [Amazon EC2 Instances](#) and [Amazon Linux AMI Instance Type Matrix](#).

Important

When you choose an instance type using the AWS Management Console, the number of **vCPU** shown for each **Instance type** is the number of YARN vcores for that instance type, not the number of EC2 vCPUs for that instance type. For more information on the number of vCPUs for each instance type, see [Amazon EC2 Instance Types](#).

Instance Class	Instance Types
General purpose	m1.medium m1.large m1.xlarge m2.xlarge m2.2xlarge m2.4xlarge m3.xlarge m3.2xlarge m4.large m4.xlarge m4.2xlarge m4.4xlarge m4.10xlarge m4.16xlarge
Compute optimized	c1.medium c1.xlarge c3.xlarge c3.2xlarge c3.4xlarge c3.8xlarge cc2.8xlarge c4.xlarge c4.2xlarge c4.4xlarge c4.8xlarge
Memory optimized	r3.xlarge r3.2xlarge r3.4xlarge r3.8xlarge r4.xlarge r4.2xlarge r4.4xlarge r4.8xlarge r4.16xlarge cr1.8xlarge
Storage optimized	hi1.4xlarge hs1.2xlarge hs1.4xlarge hs1.8xlarge i2.xlarge i2.2xlarge i2.4xlarge i2.8xlarge d2.xlarge d2.2xlarge d2.4xlarge d2.8xlarge
GPU instances	g2.2xlarge cg1.4xlarge

Note

M4 and C4 instance types are only supported on Amazon EMR releases 3.10.0 and 4.x or greater, and all new releases launch clusters with EBS-backed storage for root volumes on all instance types.

Virtualization types

The following table shows the virtualization type for each instance family used in Amazon EMR. For more information, see [Linux AMI Virtualization Types](#) in the *Amazon EC2 User Guide for Linux Instances*.

Instance Family	Virtualization Types
m1	PVM
m2	PVM
m3	PVM
m4	HVM
c1	PVM
c3	PVM
c4	HVM
cc1	HVM
cc2	HVM
cg1	HVM
cr1	HVM
g2	HVM
hi1	PVM
hs1	PVM
i2	HVM
r3	HVM
r4	HVM
d2	HVM

Enhanced Networking

Some instance types support enhanced networking provided that the instance type selected uses the HVM virtualization type, as indicated in the previous table. For more information about enhanced networking, see the following:

- [Linux AMI Virtualization Types](#)
- [Enhanced Networking on Linux](#)
- [Placement Groups](#)

Instance Purchasing Options

In addition to the instance type, you choose a purchasing option for instances. You can choose to use On-Demand Instances, Spot Instances, or both. If you choose to use the uniform instance group configuration, the instance type and purchasing option apply to all EC2 instances in each instance group, and you can only specify these options for an instance group when you create it. If you choose to use the instance fleet configuration, you can change purchasing options after an instance group is created, and you can mix purchasing options to fulfill a target capacity that you specify. For more information about these configurations, see [Create a Cluster with Instance Fleets or Uniform Instance Groups \(p. 85\)](#).

Important

When you choose an instance type using the AWS Management Console, the number of **vCPU** shown for each **Instance type** is the number of YARN vcores for that instance type, not the number of EC2 vCPUs for that instance type. For more information on the number of vCPUs for each instance type, see [Amazon EC2 Instance Types](#).

On-Demand Instances

When you specify the On-Demand purchasing option for EC2 instances in Amazon EMR, you designate that you want to pay for compute capacity by the hour. Optionally, you can have these On-Demand Instances use the Reserved Instance or Dedicated Instance purchasing options. With Reserved Instances, you make a one-time payment for an instance to reserve capacity, which can offer significant savings over Dedicated Instances. Dedicated Instances are physically isolated at the host hardware level from instances that belong to other AWS accounts. For more information about Amazon EC2 purchasing options and pricing, see [Instance Purchasing Options](#) in the *Amazon EC2 User Guide for Linux Instances*.

Using Reserved Instances

To use Reserved Instances in Amazon EMR, you use Amazon EC2 to purchase the Reserved Instance and specify the parameters of the reservation, including the scope of the reservation as applying to either a region or an Availability Zone. For more information, see [Amazon EC2 Reserved Instances](#) and [Buying Reserved Instances](#) in the *Amazon EC2 User Guide for Linux Instances*. After you've purchased a Reserved Instance, Amazon EMR uses the Reserved Instance when a cluster launches and all of the following conditions are true:

- An On-Demand Instance is specified in the cluster configuration that matches the Reserved Instance specification
- The cluster is launched within the scope of the instance reservation (the Availability Zone or region)
- The Reserved Instance capacity is still available

For example, let's say you purchase one `m1.small` Reserved Instance with the instance reservation scoped to the US-East region. You then launch an EMR cluster in US-East that uses two `m1.small` instances. The first instance is billed at the Reserved Instance rate and the other is billed at the On-Demand rate. Reserved Instance capacity is used before any On-Demand Instances are created.

Using Dedicated Instances

To use Dedicated Instances, you purchase Dedicated Instances using Amazon EC2 and then create a VPC with the **Dedicated** tenancy attribute. Within Amazon EMR, you then specify that a cluster should launch in this VPC. Any On-Demand Instances in the cluster that conform to your Dedicated Instance specification use available Dedicated Instances when the cluster launches.

Note

Amazon EMR does not support setting the `dedicated` attribute on individual instances.

Spot Instances

Spot Instances in Amazon EMR provide an option for you to purchase Amazon EC2 instance capacity at a reduced cost as compared to On-Demand purchasing. The disadvantage of using Spot Instances is that

instances may terminate unpredictably as prices fluctuate. When you create a cluster with instance fleets, you have the option to use a *defined duration* (also known as a Spot block) which provides a greater degree of predictability. Spot instances terminate at the end of the duration, but will not be interrupted before the duration expires. This topic describes how Spot Instances work with Amazon EMR. For further details about Spot Instances, see [Spot Instances](#) in the *Amazon EC2 User Guide for Linux Instances*.

When Amazon EC2 has unused capacity, it offers EC2 instances at a reduced cost, called the *Spot price*. This price fluctuates based on availability and demand, and is established by region and Availability Zone. For current pricing, you can see [Amazon EC2 Spot Instances Pricing](#). When you create and configure a cluster, you specify network options that ultimately determine the Availability Zone where your cluster launches. For more information, see [Plan and Configure Networking \(p. 73\)](#). When you configure a cluster with instance groups or instance fleets, you can designate the Spot purchasing option for each node type, along with the maximum *bid price* you're willing to pay for each EC2 instance type.

When the Spot price in the cluster's Availability Zone is below the bid price specified for that instance type, instances launch. While instances run, you're charged at the current Spot price (not your bid price). If you manually start and terminate Spot Instances, partial hours are billed as full hours. On the other hand, if instances are terminated because the Spot price rises above the bid price, you are not charged either the Amazon EC2 or Amazon EMR charges for the partial hour.

Tip

You can see the real-time Spot price in the console when you mouse over the information tooltip next to **Bid Price**. The prices for each Availability Zone in the selected region are displayed. The lowest prices are in the green-colored rows. The Spot price may change due to fluctuations in supply and demand in an Availability Zone, but the Amazon EMR rate remains fixed. Because of fluctuating Spot prices between Availability Zones, selecting the Availability Zone with the lowest initial price might not result in the lowest price for the life of the cluster. For optimal results, study the history of Availability Zone pricing before choosing. For more information, see [Spot Instance Pricing History](#) in the *Amazon EC2 User Guide for Linux Instances*.

Spot Instance options depend on whether you use uniform instance groups or instance fleets in your cluster configuration.

Spot Instances in uniform instance groups

When you use Spot Instances in a uniform instance group, all instances in an instance group must be Spot Instances. You specify a single subnet or Availability Zone for the cluster. For each instance group, you specify a single Spot Instance type and a bid price. Spot Instances of that type launch if the Spot price in the cluster's region and Availability Zone is below the bid price. Instances terminate if the spot price is above your bid price. You set the bid price only when you configure an instance group. It can't be changed later. For more information, see [Create a Cluster with Instance Fleets or Uniform Instance Groups \(p. 85\)](#).

Spot Instances in instance fleets

When you use the instance fleets configuration, additional options give you more control over how Spot Instances launch and terminate. Fundamentally, instance fleets use a different method than uniform instance groups to launch instances. The way it works is you establish a *target capacity* for Spot Instances (and On-Demand Instances) and up to five instance types. You can also specify a *weighted capacity* for each instance type or use the vCPU (YARN vcores) of the instance type as weighted capacity. This weighted capacity counts toward your target capacity when an instance of that type is provisioned. Amazon EMR provisions instances with both purchasing options until the target capacity for each target is fulfilled. In addition, you can define a range of Availability Zones for Amazon EMR to choose from when launching instances. You also provide additional Spot options for each fleet, including a provisioning timeout and, optionally, a defined duration. For more information, see [Configure Instance Fleets \(p. 86\)](#).

Instance Store and Amazon EBS

There are two types of storage volumes available for EC2 instances: Amazon EBS volumes and the instance store. With Amazon EMR, both types of storage are ephemeral, meaning the data on the volumes does not persist through instance termination. This ephemeral storage is ideal for temporary storage of information that changes frequently, such as HDFS data, as well as buffers, caches, scratch data, and other temporary content that some applications may "spill" to the local file system. Although this ephemeral storage is used for HDFS, EMRFS can help ensure that there is a persistent "source of truth" for data stored in Amazon S3. For more information about EMRFS, see [EMR File System \(EMRFS\)](#) (p. 28). For more information about EC2 instance store, see [Amazon EC2 Instance Store](#) in the *Amazon EC2 User Guide for Linux Instances*.

Note

Amazon EBS storage is available for releases 4.0 or greater.

Whether the root device volume uses the instance store or an EBS volume depends on the AMI. Some AMIs are backed by Amazon EC2 instance store, and some are backed by Amazon EBS. For more information, see [Amazon EC2 Root Device Volume](#) in the *Amazon EC2 User Guide for Linux Instances*. Amazon EMR automatically attaches an EBS General Purpose SSD (gp2) 10 GB volume as the root device for its AMIs to enhance performance. The EBS costs are pro-rated by the hour based on the monthly EBS charges for gp2 volumes in the region where the cluster runs. For example, the EBS cost per hour for the root volume for each node in your EMR cluster in a region that charges \$0.10/GB-Month would be approximately \$0.00139 per hour (\$0.10/GB-month divided by 30 days divided by 24h times 10 GB).

When you configure instance types in Amazon EMR, you can specify additional EBS volumes, which adds capacity beyond the instance store (if present) and the default EBS volume. Amazon EBS provides the following volume types: General Purpose (SSD), Provisioned IOPS (SSD), Throughput Optimized (HDD), Cold (HDD), and Magnetic. They differ in performance characteristics and price, so you can tailor your storage based on the analytic and business needs of your applications. For example, some applications may have a need to spill to disk while others can safely work in-memory or using Amazon S3.

Amazon EBS works differently within Amazon EMR than it does with regular Amazon EC2 instances. For example, EBS volumes attached to EMR clusters are ephemeral: the volumes are deleted upon cluster and instance termination (for example, when shrinking instance groups), so it is important to not expect data persistence. Although the data is ephemeral on these volumes, it is possible that data in HDFS may be replicated depending on the number and specialization of nodes in the cluster. When you add EBS volumes, these are mounted as additional volumes. They are not a part of the boot volume and represent additional storage. YARN is configured to use all the additional volumes, but you are responsible for allocating the additional volumes as local storage (for local log files for example).

You can only attach EBS volumes to instances at cluster startup time unless you add an extra task node instance group, at which time you can add EBS volumes. If an instance in an EMR cluster fails, then both the instance and attached EBS volumes are replaced as new. Consequently, if you manually detach an EBS volume, Amazon EMR treats that as a failure and replaces both instance storage (if applicable) and the volume stores.

Other caveats for using Amazon EBS with EMR clusters are:

- You cannot snapshot an EBS volume used with Amazon EMR.
- EBS-encrypted volumes are not supported.
- If you apply tags using the Amazon EMR webservice API, those operations are applied to EBS volumes.
- There is a limit of 25 volumes per instance.

Plan and Configure Networking

There may be two network platform options you can choose for your cluster: **EC2-Classic** or **EC2-VPC**. In EC2-Classic, your instances run in a single, flat network that you share with other customers. EC2-Classic

is available only with certain accounts in certain regions. For more information, see [Amazon EC2 and Amazon VPC](#) in the *Amazon EC2 User Guide for Linux Instances*. In EC2-VPC, your cluster uses Amazon Virtual Private Cloud (Amazon VPC), and EC2 instances run in a VPC that's logically isolated within your AWS account. Amazon VPC enables you to provision a virtual private cloud (VPC), an isolated area within AWS where you can configure a virtual network, controlling aspects such as private IP address ranges, subnets, routing tables, and network gateways.

VPC offers the following capabilities:

- **Processing sensitive data**

Launching a cluster into a VPC is similar to launching the cluster into a private network with additional tools, such as routing tables and network ACLs, to define who has access to the network. If you are processing sensitive data in your cluster, you may want the additional access control that launching your cluster into a VPC provides. Furthermore, you can choose to launch your resources into a private subnet where none of those resources has direct Internet connectivity.

- **Accessing resources on an internal network**

If your data source is located in a private network, it may be impractical or undesirable to upload that data to AWS for import into Amazon EMR, either because of the amount of data to transfer or because of the sensitive nature of the data. Instead, you can launch the cluster into a VPC and connect your data center to your VPC through a VPN connection, enabling the cluster to access resources on your internal network. For example, if you have an Oracle database in your data center, launching your cluster into a VPC connected to that network by VPN makes it possible for the cluster to access the Oracle database.

Public and private subnets

You can launch EMR clusters in both public and private VPC subnets. This means you do not need Internet connectivity to run an EMR cluster; however, you may need to configure network address translation (NAT) and VPN gateways to access services or resources located outside of the VPC, for example in a corporate intranet or public AWS service endpoints like AWS Key Management Service.

Important

Amazon EMR only supports launching clusters in private subnets in releases 4.2 or greater.

For more information about Amazon VPC, see the [Amazon VPC User Guide](#).

Topics

- [Amazon VPC Options \(p. 74\)](#)
- [Set Up a VPC to Host Clusters \(p. 79\)](#)
- [Launch Clusters into a VPC \(p. 82\)](#)
- [Restrict Permissions to a VPC Using IAM \(p. 84\)](#)
- [Minimum Amazon S3 Policy for Private Subnet \(p. 84\)](#)
- [Learn More \(p. 85\)](#)

Amazon VPC Options

When launching an EMR cluster within a VPC, you can launch it within either a public or private subnet. There are slight, notable differences in configuration, depending on the subnet type you choose for a cluster.

Public Subnets

EMR clusters in a public subnet require a connected Internet gateway. This is because Amazon EMR clusters must access AWS services and Amazon EMR. If a service, such as Amazon S3, provides the ability

to create a VPC endpoint, you can access those services using the endpoint instead of accessing a public endpoint through an Internet gateway. Additionally, Amazon EMR cannot communicate with clusters in public subnets through a network address translation (NAT) device. An Internet gateway is required for this purpose but you can still use a NAT instance or gateway for other traffic in more complex scenarios.

If you have additional AWS resources that you do not want connected to the Internet gateway, you can launch those components in a private subnet that you create within your VPC.

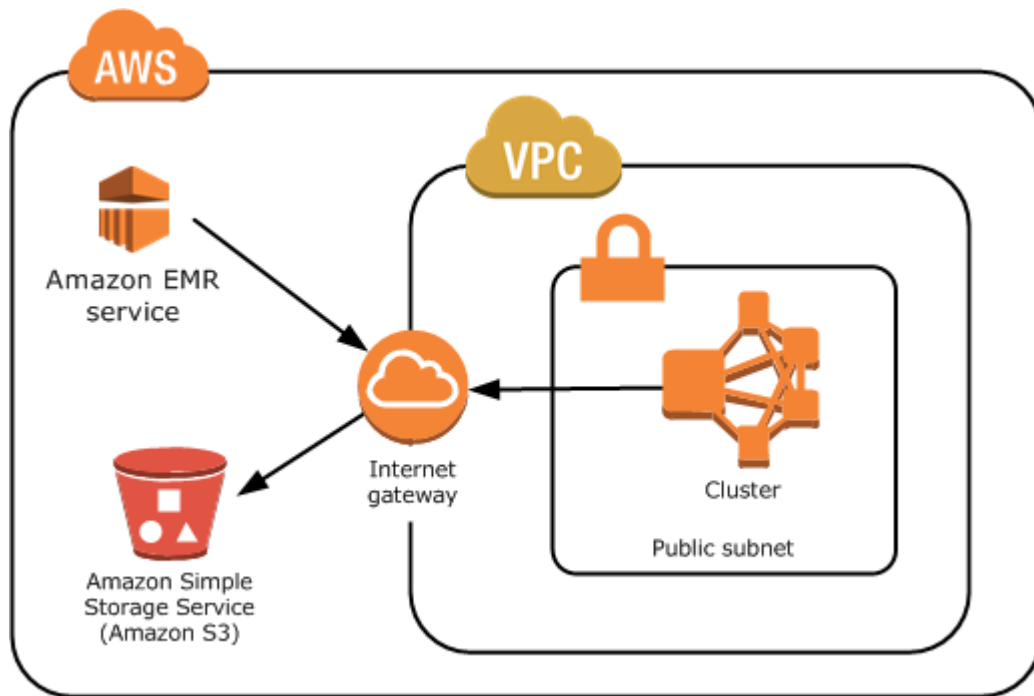
Clusters running in a public subnet use two security groups, ElasticMapReduce-master and ElasticMapReduce-slave, which control access to the master and slave instance groups, respectively.

Public Subnet Security Groups

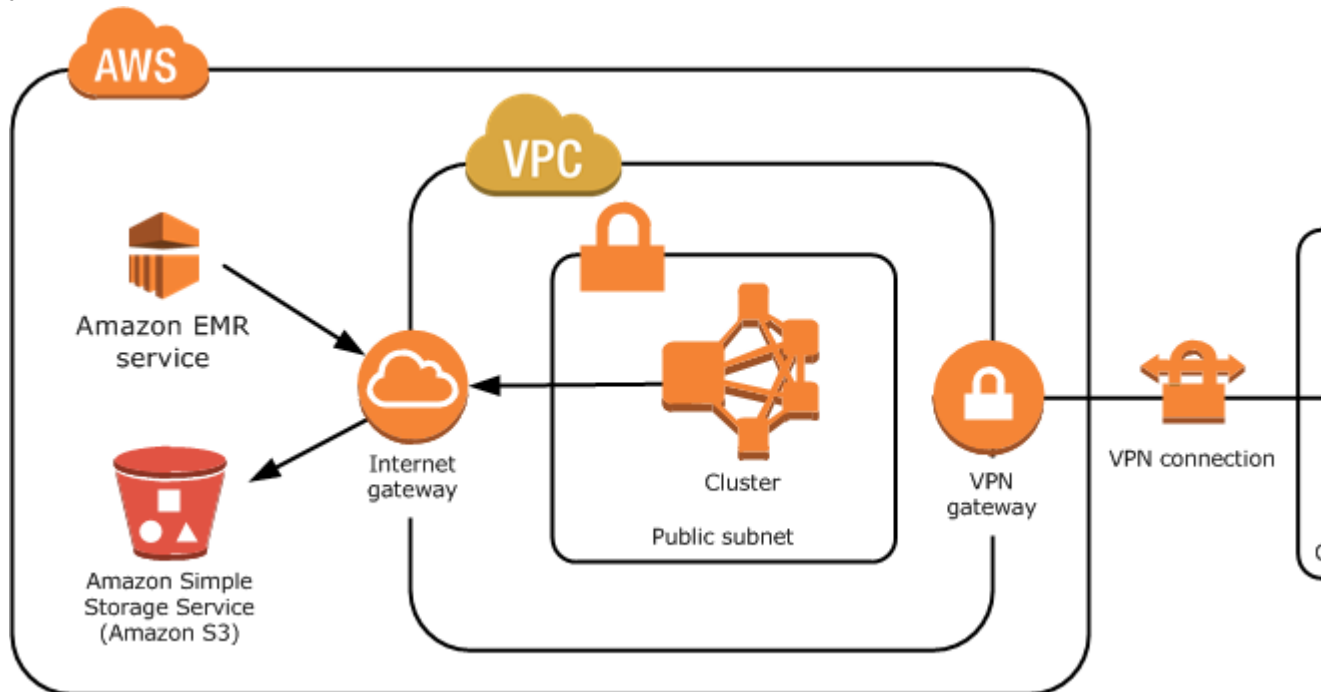
Security Group Name	Description	Open Inbound Ports	Open Outbound Ports
ElasticMapReduce-master	Security group for master instance groups of clusters in a public subnet.	TCP 0-65535 8443 22 UDP 0-65535	All
ElasticMapReduce-slave	Security group for slave instance groups (containing core and task nodes) of clusters in a public subnet.	TCP 0-65535 UDP 0-65535	All

The master instance group contains the master node while a slave group contains both task and core nodes of the cluster. All instances in a cluster connect to Amazon S3 through either a VPC endpoint or Internet gateway. Other AWS services which do not currently support VPC endpoints use only an Internet gateway.

The following diagram shows how an Amazon EMR cluster runs in a VPC using a public subnet. The cluster is able to connect to other AWS resources, such as Amazon S3 buckets, through the Internet gateway.



The following diagram shows how to set up a VPC so that a cluster in the VPC can access resources in your own network, such as an Oracle database.



Private Subnets

Private subnets allow you to launch AWS resources without requiring the subnet to have an attached Internet gateway. This might be useful, for example, in an application that uses these private resources in the back end. Those resources can then initiate outbound traffic using a NAT instance located in another

subnet that has an Internet gateway attached. For more information about this scenario, see [Scenario 2: VPC with Public and Private Subnets \(NAT\)](#).

Important

Amazon EMR only supports launching clusters in private subnets in releases 4.2 or later.

The following are differences from public subnets:

- To access AWS services that do not provide a VPC endpoint, you still must use a NAT instance or an Internet gateway. Currently, the only service supported with a VPC endpoint is Amazon S3.
- At a minimum, you must provide a route to the Amazon EMR service logs bucket and Amazon Linux repository in Amazon S3. For more information, see [Minimum Amazon S3 Policy for Private Subnet \(p. 84\)](#)
- If you use EMRFS features, you need to have an Amazon S3 VPC endpoint and a route from your private subnet to DynamoDB.
- Debugging only works if you provide a route from your private subnet to a public Amazon SQS endpoint.
- Creating a private subnet configuration with a NAT instance or gateway in a public subnet is only supported using the AWS Management Console. The easiest way to add and configure NAT instances and Amazon S3 VPC endpoints for EMR clusters is to use the **VPC Subnets List** page in the Amazon EMR console. To configure NAT gateways, follow the procedures outlined in the section called *NAT Gateways* in the [Amazon Virtual Private Cloud User Guide](#).
- You cannot change a subnet with an existing EMR cluster from public to private or vice versa. To locate an EMR cluster within a private subnet, the cluster must be started in that private subnet.

Amazon EMR creates different security groups for the clusters in a private subnet: ElasticMapReduce-Master-Private, ElasticMapReduce-Slave-Private, and ElasticMapReduce-ServiceAccess.

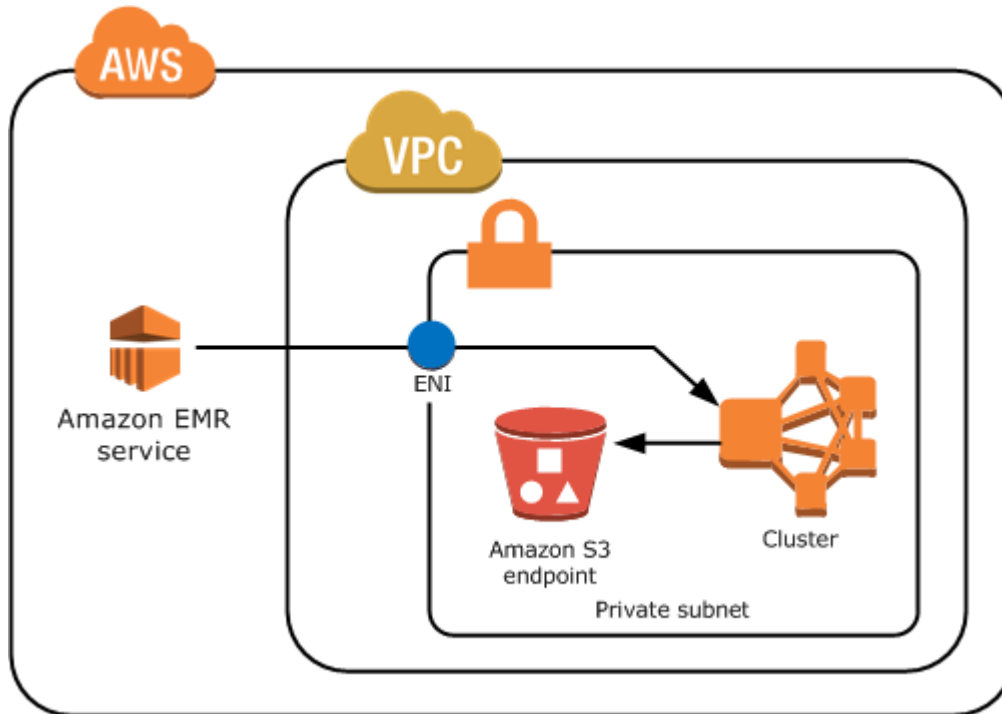
Private Subnet Security Groups

Security Group Name	Description	Open Inbound Ports	Open Outbound Ports
ElasticMapReduce-Master-Private	Security group for master instance groups of clusters in a private subnet.	TCP 0-65535 8443 UDP 0-65535	All
ElasticMapReduce-Slave-Private	Security group for slave instance groups (containing core and task nodes) of clusters in a private subnet.	TCP 0-65535 8443 UDP 0-65535	All
ElasticMapReduce-ServiceAccess	Security group for Amazon EMR-managed ENI resources used to allow communication from the web service to the cluster. The ENI is owned by you but	N/A	8443

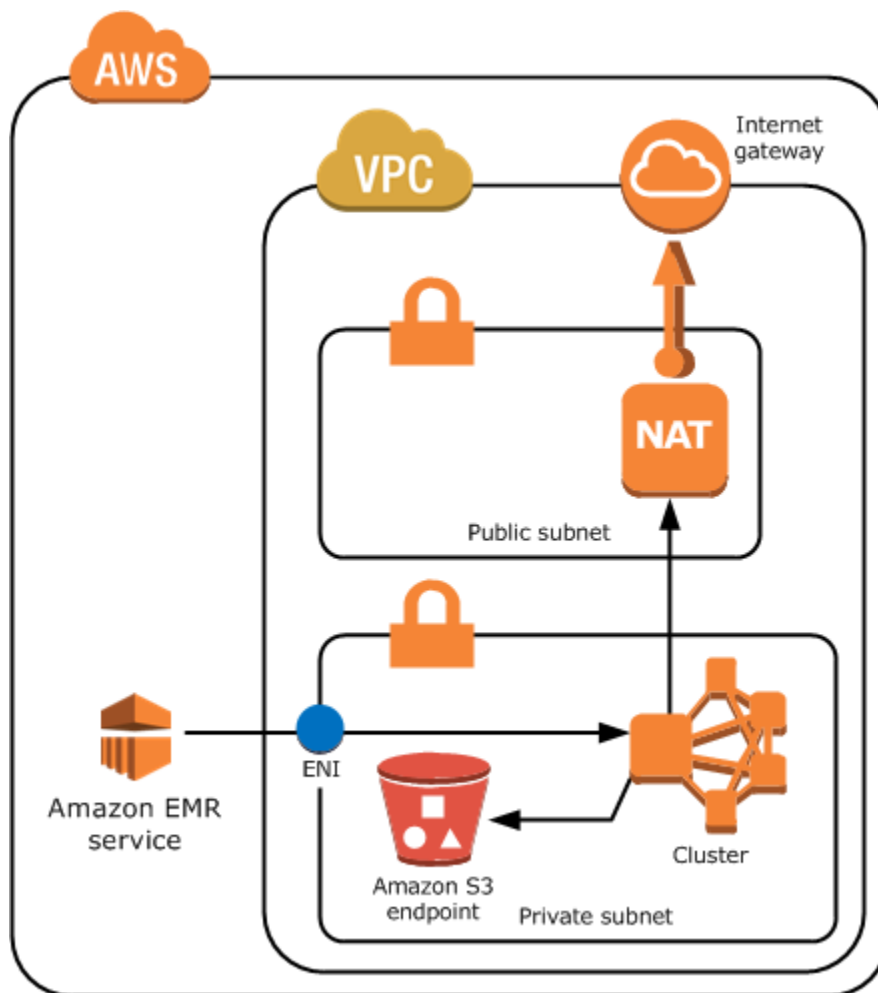
Security Group Name	Description	Open Inbound Ports	Open Outbound Ports
	managed by Amazon EMR.		

For a complete listing of NACLs of your cluster, click on the hyperlinked **Security groups for Master** and **Security groups for Core & Task** in the Amazon EMR Console Cluster Details page.

The following image shows how an EMR cluster is configured within a private subnet. The only communication outside the subnet is to Amazon EMR.



The following image shows a sample configuration for an EMR cluster within a private subnet connected to a NAT instance residing in a public subnet.



Set Up a VPC to Host Clusters

Before you can launch clusters in a VPC, you must create a VPC, and a subnet. For public subnets, you must create an Internet gateway and attach it to the subnet. The following instructions describe how to create a VPC capable of hosting Amazon EMR clusters.

To create a subnet to run Amazon EMR clusters

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation bar, select the region in which to run your cluster.
3. Choose **Start VPC Wizard**.
4. Choose the VPC configuration by selecting one of the following options:
 - **VPC with a Single Public Subnet**—Select this option if the data used in the cluster is available on the Internet (for example, in Amazon S3 or Amazon RDS).
 - **VPC with Public and Private subnets and Hardware VPN Access**—Select this option to use a private subnet or if data for your application is stored in your own network (for example, in an Oracle database). This option also allows you to include public subnets within the same VPC as private subnets.
5. Confirm the VPC settings. The images show both single public and private and public scenarios.

Step 2: VPC with a Single Public Subnet

IP CIDR block:* (65531 IP addresses available)

VPC name:

Public subnet:* (251 IP addresses available)

Availability Zone:*

Subnet name:

You can add more subnets after AWS creates the VPC.

Enable DNS hostnames:* ☒ Yes ☐ No

Hardware tenancy:*

[Cancel and Exit](#)

[Back](#)

[Create VPC](#)

Step 2: VPC with Public and Private Subnets

IP CIDR block:* (65531 IP addresses available)

VPC name:

Public subnet:* (251 IP addresses available)

Availability Zone:*

Public subnet name:

Private subnet:* (251 IP addresses available)

Availability Zone:*

Private subnet name:

You can add more subnets after AWS creates the VPC.

Specify the details of your NAT instance ([Instance rates apply](#)).

Instance type:*

Key pair name:

Add endpoints for S3 to your subnets

Subnet:

Enable DNS hostnames:* ☒ Yes ☐ No

Hardware tenancy:*

- To work with Amazon EMR, the VPC with a public subnet must have both an Internet gateway and a subnet.

For a VPC in a private subnet, your master and slave nodes must at least have a route to Amazon EMR through the ENI. In the console, this is automatically configured for you.

- Use a private IP address space for your VPC to ensure proper DNS hostname resolution; otherwise, you may experience Amazon EMR cluster failures. This includes the following IP address ranges:
 - 10.0.0.0 - 10.255.255.255
 - 172.16.0.0 - 172.31.255.255
 - 192.168.0.0 - 192.168.255.255

- Choose **Use a NAT instance instead** and select options as appropriate.
- Optionally choose to **Add endpoints for S3 to your subnets**.
- Verify that **Enable DNS hostnames** is checked. You have the option to enable DNS hostnames when you create the VPC. To change the setting of DNS hostnames, select your VPC in the VPC list, then choose **Edit** in the details pane. To create a DNS entry that does not include a domain name, create a value for **DHCP Options Set**, and then associate it with your VPC. You cannot edit the domain name using the console after the DNS option set has been created.

For more information, see [Using DNS with Your VPC](#).

- It is a best practice with Hadoop and related applications to ensure resolution of the fully qualified domain name (FQDN) for nodes. To ensure proper DNS resolution, configure a VPC that includes a DHCP options set whose parameters are set to the following values:

- **domain-name** = `ec2.internal`

Use `ec2.internal` if your region is US East (N. Virginia). For other regions, use `region-name.compute.internal`. For examples in `us-west-2`, use `us-west-2.compute.internal`. For the AWS GovCloud (US) region, use `us-gov-west-1.compute.internal`.

- **domain-name-servers** = `AmazonProvidedDNS`

For more information, see [DHCP Options Sets](#) in the *Amazon VPC User Guide*.

6. Choose **Create VPC**. If you are creating a NAT instance, it may take a few minutes for this to complete.

After the VPC is created, go to the **Subnets** page and note the identifier of one of the subnets of your VPC. You use this information when you launch the EMR cluster into the VPC.

Launch Clusters into a VPC

After you have a subnet that is configured to host Amazon EMR clusters, launch the cluster in that subnet by specifying the associated subnet identifier when creating the cluster.

Note

Amazon EMR supports private subnets in release versions 4.2 and above.

When the cluster is launched, Amazon EMR adds security groups based on whether the cluster is launching into VPC private or public subnets. All security groups allow ingress at port 8443 to communicate to the Amazon EMR service, but IP address ranges vary for public and private subnets. Amazon EMR manages all of these security groups, and may need to add additional IP addresses to the AWS range over time.

In public subnets, Amazon EMR creates ElasticMapReduce-slave and ElasticMapReduce-master for the slave and master instance groups, respectively. By default, the ElasticMapReduce-master security group allows inbound SSH connections while the ElasticMapReduce-slave group does not. Both master and slave security groups allow inbound traffic on port 8443 from the AWS public IP range. If you require SSH access for slave (core and task) nodes, you can add a rule to the ElasticMapReduce-slave security group or use SSH agent forwarding.

Other security groups and rules are required when launching clusters in a private subnet. This is to ensure that the service can still manage those resources while they are private. The additional security groups are: *ElasticMapReduce-Master-Private*, *ElasticMapReduce-Slave-Private*. The security group for the ENI is of the form *ElasticMapReduce-ServiceAccess*. Inbound traffic on port 8443 is open to allow contact to the Amazon EMR web service. Outbound traffic on port 80 and 443 should be allowed so that the cluster can communicate back to the service. Furthermore, inbound and output ephemeral ports should be open in your network ACLs.

For more information about modifying security group rules, see [Adding Rules to a Security Group](#) in the *Amazon EC2 User Guide for Linux Instances*. For more information about connecting to instances in your VPC, see [Securely connect to Linux instances running in a private Amazon VPC](#).

To manage the cluster on a VPC, Amazon EMR attaches a network device to the master node and manages it through this device. You can view this device using the Amazon EC2 API action [DescribeInstances](#). If you modify this device in any way, the cluster may fail.

To launch a cluster into a VPC using the Amazon EMR console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**.
3. Choose **Go to advanced options**.
4. In the **Hardware Configuration** section, for **Network**, select the ID of a VPC network that you created previously.
5. For **EC2 Subnet**, select the ID of a subnet that you created previously.
 - a. If your private subnet is properly configured with NAT instance and S3 endpoint options, it displays **(EMR Ready)** above the subnet names and identifiers.
 - b. If your private subnet does not have a NAT instance and/or S3 endpoint, you can configure this by choosing **Add S3 endpoint and NAT instance**, **Add S3 endpoint**, or **Add NAT instance**. Select the desired options for your NAT instance and S3 endpoint and choose **Configure**.

Important

In order to create a NAT instance from the Amazon EMR, you need `ec2:CreateRoute`, `ec2:RevokeSecurityGroupEgress`, `ec2:AuthorizeSecurityGroupEgress`, `cloudformation:DescribeStackEvents` and `cloudformation:CreateStack` permissions.

Note

There is an additional cost for launching an EC2 instance for your NAT device.

6. Proceed with creating the cluster.

To launch a cluster into a VPC using the AWS CLI

Note

The AWS CLI does not provide a way to create a NAT instance automatically and connect it to your private subnet. However, to create a S3 endpoint in your subnet, you can use the Amazon VPC CLI commands. Use the console to create NAT instances and launch clusters in a private subnet.

After your VPC is configured, you can launch EMR clusters in it by using the `create-cluster` subcommand with the `--ec2-attributes` parameter. Use the `--ec2-attributes` parameter to specify the VPC subnet for your cluster.

- To create a cluster in a specific subnet, type the following command, replace `myKey` with the name of your EC2 key pair, and replace `77XXXX03` with your subnet ID.

```
aws emr create-cluster --name "Test cluster" --release-label emr-4.2.0 --
applications Name=Hadoop Name=Hive Name=Pig --use-default-roles --ec2-attributes
KeyName=myKey,SubnetId=subnet-77XXXX03 --instance-type m3.xlarge --instance-count 3
```

When you specify the instance count without using the `--instance-groups` parameter, a single master node is launched, and the remaining instances are launched as core nodes. All nodes use the instance type specified in the command.

Note

If you have not previously created the default Amazon EMR service role and EC2 instance profile, type `aws emr create-default-roles` to create them before typing the `create-cluster` subcommand.

For more information about using Amazon EMR commands in the AWS CLI, see the [AWS CLI](#).

Restrict Permissions to a VPC Using IAM

When you launch a cluster into a VPC, you can use AWS Identity and Access Management (IAM) to control access to clusters and restrict actions using policies, just as you would with clusters launched into EC2-Classic. For more information about IAM, see [IAM User Guide](#).

You can also use IAM to control who can create and administer subnets. For more information about administering policies and actions in Amazon EC2 and Amazon VPC, see [IAM Policies for Amazon EC2](#) in the *Amazon EC2 User Guide for Linux Instances*.

By default, all IAM users can see all of the subnets for the account, and any user can launch a cluster in any subnet.

You can limit access to the ability to administer the subnet, while still allowing users to launch clusters into subnets. To do so, create one user account that has permissions to create and configure subnets and a second user account that can launch clusters but which can't modify Amazon VPC settings.

Minimum Amazon S3 Policy for Private Subnet

For private subnets, at a minimum you must provide the ability for Amazon EMR to access Amazon Linux repositories and Amazon EMR service support log buckets. The following policy provides these permissions. Replace *MyRegion* with the region where your log buckets reside, for example `us-east-1`:

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Sid": "AmazonLinuxAMIRepositoryAccess",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": [
        "arn:aws:s3:::packages.*.amazonaws.com/*",
        "arn:aws:s3:::repo.*.amazonaws.com/*"
      ]
    },
    {
      "Sid": "AccessToEMRLogBucketsForSupport",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "s3:Put*",
        "s3:Get*",
        "s3:Create*",
        "s3:Abort*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::aws157-logs-prod-MyRegion/*",
        "arn:aws:s3:::aws157-logs-prod/*"
      ]
    }
  ]
}
```

}

Learn More

Private subnets in a VPC

- [Scenario 2: VPC with Public and Private Subnets \(NAT\)](#)
- [NAT Instances](#)
- [High Availability for Amazon VPC NAT Instances: An Example](#)

Public subnets in a VPC

- [Scenario 1: VPC with a Single Public Subnet](#)

General VPC information

- [Amazon VPC User Guide](#)
- [VPC Peering](#)
- [Using Elastic Network Interfaces with Your VPC](#)
- [Securely connect to Linux instances running in a private Amazon VPC](#)

Create a Cluster with Instance Fleets or Uniform Instance Groups

When you create a cluster and specify the configuration of the master node, core nodes, and task nodes, you have two configuration options. You can use *instance fleets* or *uniform instance groups*. The configuration option you choose applies to all nodes, it applies for the lifetime of the cluster, and instance fleets and instance groups cannot coexist in a cluster. The instance fleets configuration is available in Amazon EMR version 4.8.0 and later, excluding 5.0.x versions.

You can use the EMR console, the AWS CLI, or the EMR API to create clusters with either configuration. When you use the `create-cluster` command from the AWS CLI, you use either the `--instance-fleets` parameters to create the cluster using instance fleets or, alternatively, you use the `--instance-groups` parameters to create it using uniform instance groups.

The same is true using the EMR API. You use either the `InstanceGroups` configuration to specify an array of `InstanceGroupConfig` objects, or you use the `InstanceFleets` configuration to specify an array of `InstanceFleetConfig` objects.

In the EMR console, if you use the default **Quick Options** settings when you create a cluster, Amazon EMR applies the uniform instance groups configuration to the cluster and uses On-Demand Instances. To use Spot Instances with uniform instance groups, or to configure instance fleets and other customizations, choose **Advanced Options**.

Tip

To quickly and easily replicate a cluster you have already created, Amazon EMR gives you two options in the console. You can clone the cluster or generate a `create cluster` CLI command. First, choose **Cluster list** and then choose the cluster you want to replicate. Choose **AWS CLI export** to have Amazon EMR generate the equivalent `create cluster` CLI command for the cluster, which you can then copy and paste. Choose the **Clone** button to have Amazon EMR replicate your console setup. Amazon EMR presents you with the last step of the **Advanced Options** to confirm the cluster's configuration. You can either choose **Create cluster** to create

the new cluster (with the same name and a different cluster ID), or you can choose **Previous** to go back and change settings.

Topics

- [Configure Instance Fleets \(p. 86\)](#)
- [Configure Uniform Instance Groups \(p. 93\)](#)
- [Cluster Configuration Guidelines \(p. 95\)](#)

Configure Instance Fleets

The instance fleets configuration for a cluster offers the widest variety of provisioning options for EC2 instances. You can specify up to five instance types per fleet, and those instance types can be provisioned using both On-Demand and Spot purchasing options. You can also select multiple Availability Zones, specify different maximum bid prices for each instance, and choose additional Spot options for each instance fleet. Amazon EMR uses the options you specify to more intelligently and quickly provision capacity when the cluster launches. Also, while the cluster is running, if Amazon EC2 reclaims a Spot Instance because of a price increase, or an instance fails, Amazon EMR can try to replace the instances with any of the instance types that you specify. This makes it easier to regain cluster capacity during a spike in Spot pricing. You can develop a flexible and elastic resourcing strategy for each node type. For example, within specific fleets, you can have a core of On-Demand capacity supplemented with less-expensive Spot capacity if available.

Note

The instance fleets configuration is available only in Amazon EMR versions 4.8.0 and later, excluding 5.0.x versions.

Instance Fleet Options

Target Capacities and Weighted Capacity

With each instance fleet, you establish a *target capacity* for On-Demand Instances and a target capacity for Spot Instances. Each of the five instance types available per instance fleet has a *weighted capacity* that you assign. When you use the console, you can choose to have the vCPU of the instance type automatically assigned as the weighted capacity. When using the CLI, you assign weighted capacities manually.

Important

When you choose an instance type using the AWS Management Console, the number of **vCPU** shown for each **Instance type** is the number of YARN vcores for that instance type, not the number of EC2 vCPUs for that instance type. For more information on the number of vCPUs for each instance type, see [Amazon EC2 Instance Types](#).

When Amazon EMR provisions a particular instance type, it can choose any mix of the five instance types you specify to fulfill these targets, and the weighted capacity of each provisioned instance counts toward the target capacity for On-Demand or Spot as appropriate.

Amazon EMR provisions instances in a fleet until the target capacities are totally fulfilled, even if this results in an overage. For example, if there are 2 units remaining to fulfill capacity, and Amazon EMR can only provision an instance with a weighted capacity of 5 units, the instance is provisioned nevertheless, and the target capacity is exceeded by 3 units.

Spot Instance Options

When you specify a target capacity for Spot Instances, you can specify a maximum bid price for each of the five instance types. Amazon EMR compares the bid price to Spot prices offered in the Availability Zones you select, provisioning Spot Instances if the bid price exceeds the Spot price.

For the fleet as a whole, when you specify a **Defined duration**, Spot Instances are not interrupted during the defined duration and terminate after it expires. Defined duration pricing applies when you select

this option. For more information, see [Amazon EC2 Spot Instances Pricing](#). If you don't specify a defined duration, instances terminate as soon as the Spot price exceeds the bid price.

For the fleet as a whole, you also define a **provisioning timeout** and the action to take when there is unfulfilled target capacity for Spot instances when the timeout expires. The timeout applies when the cluster is provisioning capacity when it is created. You can have the cluster terminate or switch to provisioning On-Demand capacity to fulfill the remaining Spot capacity.

For more information about Spot Instances, see [Spot Instances](#) in the Amazon EC2 User Guide for Linux Instances.

Multiple Subnet (Availability Zones) Options

You can specify multiple EC2 Subnets within a VPC, each corresponding to a different Availability Zone. (If you use EC2-classic, you specify Availability Zones explicitly.) Amazon EMR hunts for the best fit from among those Availability Zones, and then provisions instances in the Availability Zone that offers the best fit. Instances are always provisioned in only one Availability Zone. You can select private subnets or public subnets, but you can't mix the two, and the Subnets you specify must be within the same VPC.

Master Node Configuration

Because the master instance fleet is only a single instance, its configuration is slightly different from core and task instance fleets. You only select either On-Demand or Spot for the master instance fleet because it consists of only one instance. If you use the console to create the instance fleet, the target capacity for the purchasing option you select is set to 1. If you use the AWS CLI, always set either `TargetSpotCapacity` or `TargetOnDemandCapacity` to 1 as appropriate. You can still choose up to five instance types for the master instance fleet. However, unlike core and task instance fleets, where Amazon EMR might provision multiple instances of different types, Amazon EMR selects a single instance type to provision for the master instance fleet.

Summary of Key Features

- One instance fleet, and only one, per node type (master, core, task). Up to five EC2 instance types specified for each fleet.
- Amazon EMR chooses any or all of the five EC2 instance types to provision with both Spot and On-Demand purchasing options.
- Establish target capacities for Spot and On-Demand per instance fleet. Assign a weighted capacity to each instance type, which counts toward the target. Amazon EMR provisions instances until each target capacity is totally fulfilled.
- Choose one Subnet (Availability Zone) or a range. Amazon EMR provisions capacity in the Availability Zone that is the best fit.
- When you specify a target capacity for Spot Instances:
 - For each instance type, specify a maximum bid price, either as a dollar amount or as percentage of the On-Demand price.
 - Specify a defined duration (also known as a spot block) if desired. Spot Instances terminate only after the defined duration expires.
 - Define a timeout period for provisioning Spot Instances and the action to take if the timeout expires —terminate the cluster or switch to On-Demand.

Use the Console to Configure Instance Fleets

Use the **Advanced options** configuration in the Amazon EMR console to create a cluster using instance fleets.

To create a cluster with instance fleets using the console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.

2. Choose **Create cluster**.
3. Choose **Go to advanced options**, enter **Software Configuration** options, and then choose **Next**.
4. Choose **Instance fleets**.
5. Choose a **Network** . If you choose an Amazon VPC for **Network**, choose a single **EC2 Subnet** or CTRL + click to choose multiple EC2 Subnets. The Subnets you select must be the same type (public or private). If you choose only one, your cluster launches in that Subnet. If you choose a group, the Subnet with the best fit is selected from the group when the cluster launches.

Note

Your account and region may give you the option to choose **Launch into EC2-Classic** for **Network**. If you choose that option, choose one or more from **EC2 Availability Zones** rather than **EC2 Subnets**. For more information, see [Amazon EC2](#) and [Amazon VPC](#) in the *Amazon EC2 User Guide for Linux Instances*.

6. Within each node type row, under **Node type**, if you want to change the default name of an instance fleet, click the pencil icon and then enter a friendly name. If want to remove the **Task** instance fleet, click the X icon.
7. Under **Target capacity**, choose options according to the following guidelines:
 - Choose how you want to define the **Target capacity**. If you choose **vCPU**, the number of YARN vcores for each **Fleet instance type** is used as its weighted capacity. If you choose **Generic units**, you assign a custom number for each target capacity, and then assign a custom weighted capacity to each instance type. A field for this purpose appears for each instance you add under **Fleet instance type**.
 - For the **Master** node, select whether the instance is **On-demand** or **Spot**.
 - For the **Core** and **Task** nodes, enter target capacities for **On-demand** and **Spot**. Amazon EMR provisions the **Fleet instance types** you specify until these capacities are fulfilled.
8. Under **Fleet instance types** for each **Node type**, choose options according to the following guidelines:
 - Choose **Add/remove instance types to fleet**, and then choose up to five instance types from the list. Amazon EMR may choose to provision any mix of these instance types when it launches the cluster.
 - If a Node type is configured with a **Target capacity** for **Spot**, choose **Maximum bid price** options. You can enter your bid price as a **% of On-Demand** pricing, or you can enter a **Dollars (\$)** amount in USD.

Tip

Mouse over the information tooltip for **Maximum bid price** to see the Spot price for all Availability Zones in the current region. The lowest Spot price is in green. You can use this information to inform your **EC2 Subnet** selection.

- If you chose **Default units** for **Target capacity**, enter the weighted capacity you want to assign to each instance type in the **Each instance counts as** box.
 - To have EBS volumes attached to the instance type when it's provisioned, click the pencil next to **EBS Storage** and then enter EBS configuration options.
9. If you established a **Target capacity** for **Spot**, choose **Advanced Spot options** according to the following guidelines:
 - **Defined duration**—if left to the default, **Not set**, Spot Instances terminate as soon as the Spot price rises above the Maximum bid price, or when the cluster terminates. If you set a value, Spot Instances don't terminate until the duration has expired.

Important

If you set a **Defined duration**, special defined duration pricing applies. For pricing details, see [Amazon EC2 Spot Instances Pricing](#).

- **Provisioning timeout**—Use these settings to control what Amazon EMR does when it can't provision Spot Instances from among the **Fleet instance types** you specify. You enter a timeout

period in minutes, and then choose whether to **Terminate the cluster** or **Switch to provisioning On-Demand Instances**. If you choose to switch to On-Demand Instances, the weighted capacity of On-Demand Instances counts toward the target capacity for Spot Instances, and Amazon EMR provisions On-Demand Instances until the target capacity for Spot Instances is fulfilled.

10. Choose **Next** and then choose other cluster settings to launch the cluster.

Use the CLI to Configure Instance Fleets

- To create and launch a cluster with instance fleets, use the `create-cluster` command along with `--instance-fleet` parameters.
- To get configuration details of the instance fleets in a cluster, use the `list-instance-fleets` command.
- To make changes to the target capacity for an instance fleet, use the `modify-instance-fleet` command.
- To add a task instance fleet to a cluster that doesn't already have one, use the `add-instance-fleet` command.

Note

Linux line continuation characters (`\`) are included for readability. They can be removed or used in Linux commands. For Windows, remove them or replace with a caret (`^`).

Create a Cluster with the Instance Fleets Configuration

The following examples demonstrate `create-cluster` commands with a variety of options that you can combine.

Note

If you have not previously created the default EMR service role and EC2 instance profile, use `aws emr create-default-roles` to create them before using the `create-cluster` command.

Example: On-Demand Master, On-Demand Core with Single Instance Type, Default VPC

```
aws emr create-cluster --release-label emr-5.3.1 --service-role EMR_DefaultRole \
--ec2-attributes InstanceProfile=EMR_EC2_DefaultRole \
--instance-fleets
  InstanceFleetType=MASTER,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=m3.xlarge}']
\
  InstanceFleetType=CORE,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=m3.xlarge}']
```

Example: Spot Master, Spot Core with Single Instance Type, Default VPC

```
aws emr create-cluster --release-label emr-5.3.1 --service-role EMR_DefaultRole \
--ec2-attributes InstanceProfile=EMR_EC2_DefaultRole \
--instance-fleets
  InstanceFleetType=MASTER,TargetSpotCapacity=1,InstanceTypeConfigs=['{InstanceType=m3.xlarge,BidPrice=0.5}']
\
  InstanceFleetType=CORE,TargetSpotCapacity=1,InstanceTypeConfigs=['{InstanceType=m3.xlarge,BidPrice=0.5}']
```

Example: On-Demand Master, Mixed Core with Single Instance Type, Single EC2 Subnet

```
aws emr create-cluster --release-label emr-5.3.1 --service-role EMR_DefaultRole \
```

```
--ec2-attributes InstanceProfile=EMR_EC2_DefaultRole,SubnetIds=[ 'subnet-ab12345c' ] \
--instance-fleets
  InstanceFleetType=MASTER,TargetOnDemandCapacity=1,InstanceTypeConfigs=[ '{ InstanceType=m3.xlarge}' ]
  \
  InstanceFleetType=CORE,TargetOnDemandCapacity=2,TargetSpotCapacity=6,InstanceTypeConfigs=[ '{ InstanceType=
```

Example: On-Demand Master, Spot Core with Multiple Weighted Instance Types, Defined Duration and Timeout for Spot, Range of EC2 Subnets

```
aws emr create-cluster --release-label emr-5.3.1 --service-role EMR_DefaultRole \
--ec2-attributes InstanceProfile=EMR_EC2_DefaultRole,SubnetIds=[ 'subnet-ab12345c', 'subnet-
de67890f' ] \
--instance-fleets
  InstanceFleetType=MASTER,TargetOnDemandCapacity=1,InstanceTypeConfigs=[ '{ InstanceType=m3.xlarge}' ]
  \
  InstanceFleetType=CORE,TargetSpotCapacity=11,InstanceTypeConfigs=[ '{ InstanceType=m3.xlarge,BidPrice=0.5,
  \
  '{ InstanceType=m4.2xlarge,BidPrice=0.9,WeightedCapacity=5}' ],\
LaunchSpecifications={SpotSpecification='{TimeoutDurationMinutes=120,TimeoutAction=SWITCH_TO_ON_DEMAND}
```

Example: On-Demand Master, Mixed Core and Task with Multiple Weighted Instance Types, Defined Duration and Timeout for Core Spot Instances, Range of EC2 Subnets

```
aws emr create-cluster --release-label emr-5.3.1 --service-role EMR_DefaultRole \
--ec2-attributes InstanceProfile=EMR_EC2_DefaultRole,SubnetIds=[ 'subnet-ab12345c', 'subnet-
de67890f' ] \
--instance-fleets
  InstanceFleetType=MASTER,TargetOnDemandCapacity=1,InstanceTypeConfigs=[ '{ InstanceType=m3.xlarge}' ]
  \
  InstanceFleetType=CORE,TargetOnDemandCapacity=8,TargetSpotCapacity=6,\
  InstanceTypeConfigs=[ '{ InstanceType=m3.xlarge,BidPrice=0.5,WeightedCapacity=3}' ,\
  '{ InstanceType=m4.2xlarge,BidPrice=0.9,WeightedCapacity=5}' ],\
  LaunchSpecifications={SpotSpecification='{TimeoutDurationMinutes=120,TimeoutAction=SWITCH_TO_ON_DEMAND}
```

Example: Spot Master, No Core or Task, EBS configuration, Default VPC

```
aws emr create-cluster --release-label emr 5.3.1 --service-role EMR_DefaultRole \
--ec2-attributes InstanceProfile=EMR_EC2_DefaultRole \
--instance-fleets InstanceFleetType=MASTER,TargetSpotCapacity=1,\
LaunchSpecifications={SpotSpecification='{TimeoutDurationMinutes=60,TimeoutAction=TERMINATE_CLUSTER}',\
  \
  InstanceTypeConfigs=[ '{ InstanceType=m3.xlarge,BidPrice=0.5,\
  EbsConfiguration={EbsOptimized=true,EbsBlockDeviceConfigs=[{VolumeSpecification={VolumeType=gp2,
  \
  SizeIn GB=100}}, {VolumeSpecification={VolumeType=io1,SizeInGB=100,Iops=100},VolumesPerInstance=4}]]}' ]
```

Use a JSON Configuration File

You can configure instance fleet parameters in a JSON file, and then reference the JSON file as the sole parameter for instance fleets. For example, the following command references a JSON configuration file, my-fleet-config.json:

```
aws emr create-cluster --release-label emr-5.2.0 --servicerole EMR_DefaultRole \  
--ec2-attributes InstanceProfile=EMR_EC2_DefaultRole \  
--instance-fleets file://my-fleet-config.json
```

The my-fleet-config.json specifies master, core, and task instance fleets as shown in the following example. The core instance fleet uses a bid price as a percentage of on-demand, while the task and master instance fleets use a bid price (as a string) in USD.

```
[  
  {  
    "Name": "Masterfleet",  
    "InstanceFleetType": "MASTER",  
    "TargetSpotCapacity": 1,  
    "LaunchSpecifications": {  
      "SpotSpecification": {  
        "TimeoutDurationMinutes": 120,  
        "TimeoutAction": "SWITCH_TO_ON_DEMAND"  
      }  
    },  
    "InstanceTypeConfigs": [  
      {  
        "InstanceType": "m1.medium",  
        "BidPrice": "0.89"  
      }  
    ]  
  },  
  {  
    "Name": "Corefleet",  
    "InstanceFleetType": "CORE",  
    "TargetSpotCapacity": 1,  
    "LaunchSpecifications": {  
      "SpotSpecification": {  
        "TimeoutDurationMinutes": 120,  
        "TimeoutAction": "TERMINATE_CLUSTER"  
      }  
    },  
    "InstanceTypeConfigs": [  
      {  
        "InstanceType": "m3.xlarge",  
        "BidPriceAsPercentageOfOnDemandPrice": 100  
      }  
    ]  
  },  
  {  
    "Name": "Taskfleet",  
    "InstanceFleetType": "TASK",  
    "TargetSpotCapacity": 1,  
    "LaunchSpecifications": {  
      "SpotSpecification": {  
        "TimeoutDurationMinutes": 120,  
        "TimeoutAction": "TERMINATE_CLUSTER"  
      }  
    },  
    "InstanceTypeConfigs": [  
      {  
        "InstanceType": "m3.xlarge",  
        "BidPrice": "0.89"  
      }  
    ]  
  }  
]
```


Get Configuration Details of Instance Fleets in a Cluster

Use the `list-instance-fleets` command to get configuration details of the instance fleets in a cluster. The command takes a cluster ID as input. The following example demonstrates the command and its output for a cluster that contains a master task instance group and a core task instance group. For full response syntax, see [ListInstanceFleets](#) in the *Amazon EMR API Reference*.

```
list-instance-fleets --cluster-id 'j-12ABCDEFGHI34JK'
```

```
{
  "InstanceFleets": [
    {
      "Status": {
        "Timeline": {
          "ReadyDateTime": 1488759094.637,
          "CreationDateTime": 1488758719.817
        },
        "State": "RUNNING",
        "StateChangeReason": {
          "Message": ""
        }
      },
      "ProvisionedSpotCapacity": 6,
      "Name": "CORE",
      "InstanceFleetType": "CORE",
      "LaunchSpecifications": {
        "SpotSpecification": {
          "TimeoutDurationMinutes": 60,
          "TimeoutAction": "TERMINATE_CLUSTER"
        }
      },
      "ProvisionedOnDemandCapacity": 2,
      "InstanceTypeSpecifications": [
        {
          "BidPrice": "0.5",
          "InstanceType": "m3.xlarge",
          "WeightedCapacity": 2
        }
      ],
      "Id": "if-1ABC2DEFGHIJ3"
    },
    {
      "Status": {
        "Timeline": {
          "ReadyDateTime": 1488759058.598,
          "CreationDateTime": 1488758719.811
        },
        "State": "RUNNING",
        "StateChangeReason": {
          "Message": ""
        }
      },
      "ProvisionedSpotCapacity": 0,
      "Name": "MASTER",
      "InstanceFleetType": "MASTER",
      "ProvisionedOnDemandCapacity": 1,
      "InstanceTypeSpecifications": [
        {
          "BidPriceAsPercentageOfOnDemandPrice": 100.0,
          "InstanceType": "m3.xlarge",

```

```
        "WeightedCapacity": 1
      },
    ],
    "Id": "if-2ABC4DEFGHIJ4"
  }
]
```

Modify Target Capacities for an Instance Fleet

Use the `modify-instance-fleet` command to specify new target capacities for an instance fleet. You must specify the cluster ID and the instance fleet ID. Use the `list-instance-fleets` command to retrieve instance fleet IDs.

```
aws emr modify-instance-fleet --cluster-id 'j-12ABCDEFGHI34JK' /
--instance-fleet
InstanceFleetId='if-2ABC4DEFGHIJ4',TargetOnDemandCapacity=1,TargetSpotCapacity=1
```

Add a Task Instance Fleet to a Cluster

If a cluster has only master and core instance fleets, you can use the `add-instance-fleet` command to add a task instance fleet. You can only use this to add task instance fleets.

```
aws emr add-instance-fleet --cluster-id 'j-12ABCDEFGHI34JK' --instance-fleet
InstanceFleetType=TASK,TargetSpotCapacity=1,/
LaunchSpecifications={SpotSpecification='{TimeoutDurationMinutes=20,TimeoutAction=TERMINATE_CLUSTER}}',
InstanceTypeConfigs=[ '{InstanceType=m3.xlarge,BidPrice=0.5}' ]
```

Configure Uniform Instance Groups

With the instance groups configuration, each node type (master, core, or task) consists of the same instance type and the same purchasing option for instances: On-Demand or Spot. You specify these settings when you create an instance group and they can't be changed later. You can, however, add instances of the same type and purchasing option to core and task instance groups. You can also remove instances.

To add different instance types after a cluster is created, you can add additional task instance groups, specifying different instance types and purchasing options for each instance group. For more information, see [Scaling Cluster Resources \(p. 190\)](#).

This section covers creating a cluster with uniform instance groups. For more information about modifying an existing instance group by adding or removing instances manually or with automatic scaling, see [Manage Clusters \(p. 139\)](#).

Use the Console to Configure Uniform Instance Groups

The following procedure covers **Advanced options** when you create a cluster. Using **Quick options** also creates a cluster with the instance groups configuration. For more information about using **Quick Options**, see the Getting Started tutorial.

To create a cluster with uniform instance groups using the console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.

2. Choose **Create cluster**.
3. Choose **Go to advanced options**, enter **Software Configuration** options, and then choose **Next**.
4. In the **Hardware Configuration** screen, leave **Uniform instance groups** selected.
5. Choose the **Network**, and then choose the **EC2 Subnet** in which you want your cluster to run. For more information about VPCs and subnets, see [Plan and Configure Networking \(p. 73\)](#).

Note

Your account and region may give you the option to choose **Launch into EC2-Classical for Network**. If you choose that option, choose an **EC2 Availability Zone** rather than an **EC2 Subnet**. For more information, see [Amazon EC2 and Amazon VPC](#) in the *Amazon EC2 User Guide for Linux Instances*.

6. Within each **Node type** row:
 - Under **Node type**, if you want to change the default name of the instance group, click the pencil icon and then enter a friendly name. If want to remove the **Task** instance group, click the X icon or choose **Add task instance group** to add additional **Task** instance groups.
 - Under **Instance type** click the pencil icon and then choose the instance type you want to use for that node type.

Important

When you choose an instance type using the AWS Management Console, the number of **vCPU** shown for each **Instance type** is the number of YARN vcores for that instance type, not the number of EC2 vCPUs for that instance type. For more information on the number of vCPUs for each instance type, see [Amazon EC2 Instance Types](#).

- Under **Instance count** enter the number of instances you want to run that node type. There is only one instance in the **Master** node type.
- Under **Purchasing option** choose **On-demand**, or choose **Spot** and enter the **Maximum bid price** you are willing to pay per instance. Instances in this instance group launch when the Spot price in the Availability Zone of the **EC2 Subnet** you chose is below the **Maximum bid price**.

Tip

Mouse over the information tooltip for **Maximum bid price** to see the Spot price for all Availability Zones in the current region. The lowest Spot price is in green. You might want to use this information to inform your **EC2 Subnet** selection.

- Under Auto Scaling for Core and Task node types, click the pencil icon and then choose automatic scaling options. For more information, see [Using Automatic Scaling in Amazon EMR \(p. 191\)](#).
7. To add another task instance group to the cluster, click and configure settings for the instance group as described in the previous step.
 8. Choose **Next** and then choose other cluster settings to launch the cluster.

Use the AWS CLI to Create a Cluster with Uniform Instance Groups

To specify the instance groups configuration for a cluster using the AWS CLI, use the `create-cluster` command along with the `--instance-groups` parameter. Amazon EMR assumes the On-Demand purchasing option unless you specify the `BidPrice` argument for an instance group. For examples of `create-cluster` commands that launch uniform instance groups with On-Demand Instances and a variety of cluster options, type `aws emr create-cluster help` at the command line, or see [create-cluster](#) in the *AWS CLI reference*.

You can use the AWS CLI to create uniform instance groups in a cluster that use Spot Instances. The offered Spot price depends on Availability Zone. When you use the CLI or API, you can specify the Availability Zone either with the `AvailabilityZone` argument (if you're using an EC2-classic network) or the `SubnetID` argument of the `--ec2-attributes` parameter. The Availability Zone or Subnet you select applies to the cluster, so it's used for all instance groups. If you don't specify an Availability Zone or Subnet explicitly, Amazon EMR selects the Availability Zone with the lowest Spot price when it launches the cluster.

The following example demonstrates a `create-cluster` command that creates master, core, and two task instance groups that all use Spot Instances. Replace `myKey` with the name of your EC2 key pair.

Note

If you have not previously created the default EMR service role and EC2 instance profile, use `aws emr create-default-roles` to create them before using the `create-cluster` command.

Linux, UNIX, and Mac OS X users:

```
aws emr create-cluster --name "MySpotCluster" --release-label emr-4.0.0 \
--use-default-roles --ec2-attributes KeyName=myKey \
--instance-groups
  InstanceGroupType=MASTER,InstanceType=m3.xlarge,InstanceCount=1,BidPrice=0.25 \
InstanceGroupType=CORE,InstanceType=m3.xlarge,InstanceCount=2,BidPrice=0.03 \
InstanceGroupType=TASK,InstanceType=m3.xlarge,InstanceCount=4,BidPrice=0.03 \
InstanceGroupType=TASK,InstanceType=m3.xlarge,InstanceCount=2,BidPrice=0.04
```

Windows users:

```
aws emr create-cluster --name "Spot cluster" --release-label emr-4.0.0 --applications
Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --instance-
groups InstanceGroupType=MASTER,InstanceType=m3.xlarge,InstanceCount=1,BidPrice=0.25
InstanceGroupType=CORE,BidPrice=0.03,InstanceType=m3.xlarge,InstanceCount=2
InstanceGroupType=TASK,BidPrice=0.03,InstanceType=m3.xlarge,InstanceCount=4
InstanceGroupType=TASK,BidPrice=0.04,InstanceType=m3.xlarge,InstanceCount=2
```

Use the Java SDK to Create an Instance Group

You instantiate an `InstanceGroupConfig` object that specifies the configuration of an instance group for a cluster. To use Spot Instances, you set the `withBidPrice` and `withMarket` properties on the `InstanceGroupConfig` object. The following code shows how to define master, core, and task instance groups that run Spot Instances.

```
InstanceGroupConfig instanceGroupConfigMaster = new InstanceGroupConfig()
    .withInstanceCount(1)
    .withInstanceRole("MASTER")
    .withInstanceType("m3.xlarge")
    .withMarket("SPOT")
    .withBidPrice("0.25");

InstanceGroupConfig instanceGroupConfigCore = new InstanceGroupConfig()
    .withInstanceCount(4)
    .withInstanceRole("CORE")
    .withInstanceType("m3.xlarge")
    .withMarket("SPOT")
    .withBidPrice("0.03");

InstanceGroupConfig instanceGroupConfigTask = new InstanceGroupConfig()
    .withInstanceCount(2)
    .withInstanceRole("TASK")
    .withInstanceType("m3.xlarge")
    .withMarket("SPOT")
    .withBidPrice("0.10");
```

Cluster Configuration Guidelines

Use the guidance in this section to help you determine the instance types, purchasing options, and amount of storage to provision for each node type in an EMR cluster.

What Instance Type Should You Use?

There are several ways to add EC2 instances to your cluster. When a cluster uses the instance groups configuration, you can add instances to the core or task instance groups, or you can add task instance groups. You can add EC2 instances manually, or you can set up automatic scaling within Amazon EMR to add instances automatically based on the value of an Amazon CloudWatch metric that you specify. For more information, see [Scaling Cluster Resources \(p. 190\)](#). When a cluster uses the instance fleets configuration, you can change the target capacity for On-Demand Instances or Spot Instances as appropriate. For more information, see [Configure Instance Fleets \(p. 86\)](#).

One way to plan the instances of your cluster is to run a test cluster with a representative sample set of data and monitor the utilization of the nodes in the cluster. For more information, see [View and Monitor a Cluster \(p. 139\)](#). Another way is to calculate the capacity of the instances you are considering and compare that value against the size of your data.

In general, the master node type, which assigns tasks, doesn't require an EC2 instance with much processing power; EC2 instances for the core node type, which process tasks and store data in HDFS, need both processing power and storage capacity; EC2 instances for the task node type, which don't store data, need only processing power. For guidelines about available EC2 instances and their configuration, see [Plan and Configure EC2 Instances \(p. 68\)](#).

The following guidelines apply to most Amazon EMR clusters.

- The master node does not have large computational requirements. For most clusters of 50 or fewer nodes, consider using an m3.xlarge instance. For clusters of more than 50 nodes, consider using an m3.2xlarge.
- The computational needs of the core and task nodes depend on the type of processing your application performs. Many jobs can be run on m3.large instance types, which offer balanced performance in terms of CPU, disk space, and input/output. If your application has external dependencies that introduce delays (such as web crawling to collect data), you may be able to run the cluster on m3.xlarge instances to reduce costs while the instances are waiting for dependencies to finish. For improved performance, consider running the cluster using m3.2xlarge instances for the core and task nodes. If different phases of your cluster have different capacity needs, you can start with a small number of core nodes and increase or decrease the number of task nodes to meet your job flow's varying capacity requirements.
- Most Amazon EMR clusters can run on standard EC2 instance types such as m3.xlarge and m3.2xlarge. Computation-intensive clusters may benefit from running on High CPU instances, which have proportionally more CPU than RAM. Database and memory-caching applications may benefit from running on High Memory instances. Network-intensive and CPU-intensive applications like parsing, NLP, and machine learning may benefit from running on Cluster Compute instances, which provide proportionally high CPU resources and increased network performance.
- The amount of data you can process depends on the capacity of your core nodes and the size of your data as input, during processing, and as output. The input, intermediate, and output data sets all reside on the cluster during processing.
- By default, the total number of EC2 instances you can run on a single AWS account is 20. This means that the total number of nodes you can have in a cluster is 20. For more information about how to request that this limit be increased for your account, see [AWS Limits](#).
- In Amazon EMR, m1.small and m1.medium instances are recommended only for testing purposes and m1.small is not supported on Hadoop 2 clusters.

When Should You Use Spot Instances?

There are several scenarios in which Spot Instances are useful for running an Amazon EMR cluster.

Long-Running Clusters and Data Warehouses

If you are running a persistent Amazon EMR cluster, such as a data warehouse, that has a predictable variation in computational capacity, you can handle peak demand at lower cost with Spot Instances. You can launch your master and core instance groups as On-Demand to handle the normal capacity and launch the task instance group as Spot Instances to handle your peak load requirements.

Cost-Driven Workloads

If you are running transient clusters for which lower cost is more important than the time to completion, and losing partial work is acceptable, you can run the entire cluster (master, core, and task instance groups) as Spot Instances to benefit from the largest cost savings.

Data-Critical Workloads

If you are running a cluster for which lower cost is more important than time to completion, but losing partial work is not acceptable, launch the master and core instance groups as on-demand and supplement with one or more task instance groups of Spot Instances. Running the master and core instance groups as on-demand ensures that your data is persisted in HDFS and that the cluster is protected from termination due to Spot market fluctuations, while providing cost savings that accrue from running the task instance groups as Spot Instances.

Application Testing

When you are testing a new application in order to prepare it for launch in a production environment, you can run the entire cluster (master, core, and task instance groups) as Spot Instances to reduce your testing costs.

Choose What to Launch as Spot Instances

When you launch a cluster in Amazon EMR, you can choose to launch any or all of the instance groups (master, core, and task) as Spot Instances. Because each type of instance group plays a different role in the cluster, the implications of launching each instance group as Spot Instances vary.

When you launch an instance group either as on-demand or as Spot Instances, you cannot change its classification while the cluster is running. In order to change an On-Demand Instance group to Spot Instances or vice versa, you must terminate the cluster and launch a new one.

The following table shows launch configurations for using Spot Instances in various applications.

Project	Master Instance Group	Core Instance Group	Task Instance Groups
Long-running clusters	On-Demand	On-Demand or instance-fleet mix	Spot or instance-fleet mix
Cost-driven workloads	Spot	Spot	Spot
Data-critical workloads	On-Demand	On-Demand	Spot or instance-fleet mix
Application testing	Spot	Spot	Spot

Master Node as Spot Instance

The master node controls and directs the cluster. When it terminates, the cluster ends, so you should only launch the master node as a Spot Instance if you are running a cluster where sudden termination is acceptable. This might be the case if you are testing a new application, have a cluster that periodically

persists data to an external store such as Amazon S3, or are running a cluster where cost is more important than ensuring the cluster's completion.

When you launch the master instance group as a Spot Instance, the cluster does not start until that Spot Instance request is fulfilled. This is something to take into consideration when selecting your bid price.

You can only add a Spot Instance master node when you launch the cluster. Master nodes cannot be added or removed from a running cluster.

Typically, you would only run the master node as a Spot Instance if you are running the entire cluster (all instance groups) as Spot Instances.

Core Instance Group as Spot Instances

Core nodes process data and store information using HDFS. You typically only run core nodes as Spot Instances if you are either not running task nodes or running task nodes as Spot Instances.

When you launch the core instance group as Spot Instances, Amazon EMR waits until it can provision all of the requested core instances before launching the instance group. This means that if you request a core instance group with six nodes, the instance group does not launch if there are only five nodes available at or below your bid price. In this case, Amazon EMR continues to wait until all six core nodes are available at your Spot price or until you terminate the cluster.

You can add Spot Instance core nodes either when you launch the cluster or later to add capacity to a running cluster. You cannot shrink the size of the core instance group in a running cluster by reducing the instance count. However, it is possible to terminate an instance in the core instance group using the AWS CLI or the API. This should be done with caution. Terminating an instance in the core instance group risks data loss, and the instance is not automatically replaced.

Task Instance Groups as Spot Instances

The task nodes process data but do not hold persistent data in HDFS. If they terminate because the Spot price has risen above your bid price, no data is lost and the effect on your cluster is minimal.

When you launch one or more task instance groups as Spot Instances, Amazon EMR provisions as many task nodes as it can at your bid price. This means that if you request a task instance group with six nodes, and only five Spot Instances are available at your bid price, Amazon EMR launches the instance group with five nodes, adding the sixth later if it can.

Launching task instance groups as Spot Instances is a strategic way to expand the capacity of your cluster while minimizing costs. If you launch your master and core instance groups as On-Demand Instances, their capacity is guaranteed for the run of the cluster and you can add task instances to your task instance groups as needed to handle peak traffic or to speed up data processing.

You can add or remove task nodes using the console, the AWS CLI or the API. You can also add additional task groups using the console, the AWS CLI or the API, but you cannot remove a task group once it is created.

Calculating the Required HDFS Capacity of a Cluster

The amount of HDFS storage available to your cluster depends on these factors:

- The number of EC2 instances in the core instance group.
- The storage capacity of the EC2 instances.
- The number and size of EBS volumes attached to core nodes.
- A replication factor, which accounts for how each data block is stored in HDFS for RAID-like redundancy. By default, the replication factor is three for a cluster of 10 or more core nodes, 2 for a cluster of 4-9 core nodes, and 1 for a cluster of 3 nodes or fewer.

To calculate the HDFS capacity of a cluster, add the capacity of instance store volumes the EC2 instance types you've selected to the total volume storage you have attached with EBS and multiply the result by the number of nodes in each instance group. Divide the total by the replication factor for the cluster. For example, a cluster with 10 core nodes of type m1.large would have 850 GB of space per-instance available to HDFS: (10 nodes x 850 GB per node) / replication factor of 3. For more information on instance store volumes, see [Amazon EC2 Instance Store](#) in the *Amazon EC2 User Guide for Linux Instances*.

If the calculated HDFS capacity value is smaller than your data, you can increase the amount of HDFS storage in the following ways:

- Creating a cluster with additional EBS volumes or adding instance groups with attached EBS volumes to an existing cluster
- Adding more core nodes
- Choosing an EC2 instance type with greater storage capacity
- Using data compression
- Changing the Hadoop configuration settings to reduce the replication factor

Reducing the replication factor should be used with caution as it reduces the redundancy of HDFS data and the ability of the cluster to recover from lost or corrupted HDFS blocks.

Configure Access to the Cluster

Amazon EMR provides several ways to control access to resources:

- **IAM users and roles** provide permissions that allow users to perform actions.
- **The Amazon EMR service role and instance profile** control how Amazon EMR is able to access other AWS services.
- **Security groups** act as a virtual firewall for Amazon EMR cluster instances, controlling inbound and outbound traffic.
- **SSH keys** allow users to connect to an Amazon EMR cluster's master node.
- **System directory permissions for Hadoop** allow you to enable users other than the "hadoop user" to submit jobs to an Amazon EMR cluster.

Access control works in tandem with data encryption. A solid defense strategy includes both components. For more information about setting up data encryption, see [Data Encryption](#) in the *Amazon EMR Release Guide*.

Topics

- [Configure User Permissions Using IAM Roles \(p. 99\)](#)
- [Configure IAM Roles for Amazon EMR and Applications \(p. 106\)](#)
- [Configure Security Groups \(p. 113\)](#)
- [Create SSH Credentials for the Master Node \(p. 119\)](#)
- [Setting Permissions on the System Directory \(p. 120\)](#)

Configure User Permissions Using IAM Roles

Amazon EMR supports IAM policies. IAM is a web service that enables AWS customers to manage users and their permissions. You can use IAM to create user-based policies and attach them to users. The

policies grant or deny permissions and determine what actions a user can perform with Amazon EMR and other AWS resources. For example, you can allow a user to view EMR clusters in an AWS account but not create or delete them. In addition, you can tag EMR clusters and then use the tags to apply fine-grained permissions to users on individual clusters or a group of clusters that share the same tag.

IAM is available at no charge to all AWS account holders. You don't need to sign up for IAM. You can use IAM through the Amazon EMR console, the AWS CLI, and programmatically through the Amazon EMR API and the AWS SDKs.

IAM policies adhere to the principle of least privilege, which means that a user can't perform an action until permission is granted to do so. For more information, see the [IAM User Guide](#).

Topics

- [Amazon EMR Actions in User-Based IAM Policies](#) (p. 100)
- [Use Managed Policies for User Access](#) (p. 100)
- [Use Cluster Tags for Fine-Grained Access Control](#) (p. 102)

Amazon EMR Actions in User-Based IAM Policies

In IAM user-based policies for Amazon EMR, all Amazon EMR actions are prefixed with the lowercase `elasticmapreduce` element. You can specify the `"elasticmapreduce:*"` key, using the wildcard character (`*`), to specify all actions related to Amazon EMR, or you can allow a subset of actions, for example, `"elasticmapreduce:Describe*"`. You can also explicitly specify individual Amazon EMR actions, for example `"elasticmapreduce:DescribeCluster"`. For a complete list of Amazon EMR actions, see the API action names in the [Amazon EMR API Reference](#). Because Amazon EMR relies on other services such as Amazon EC2 and Amazon S3, users need to be allowed a subset of permissions for these services as well. For more information, see [IAM Managed Policy for Full Access](#) (p. 101).

Note

At a minimum, to access the Amazon EMR console, an IAM user needs to have an attached IAM policy that allows the following action:

```
elasticmapreduce:ListClusters
```

For more information about permissions and policies, see [Access Management](#) in the *IAM User Guide*.

Amazon EMR does not support resource-based and resource-level policies, but you can use the `Condition` element (also called the `Condition` block) to specify fine-grained access control based on cluster tags. For more information, see [Use Cluster Tags for Fine-Grained Access Control](#) (p. 102). Because Amazon EMR does not support resource-based or resource-level policies, the `Resource` element always has a wildcard value.

The easiest way to grant permissions to users is to use the *managed policies* for Amazon EMR. Managed policies also offer the benefit of being automatically updated if permission requirements change. If you need to customize policies, we recommend starting with a managed policy and then customizing privileges and conditions according to your requirements.

Use Managed Policies for User Access

The easiest way to grant full access or read-only access to required Amazon EMR actions is to use the IAM managed policies for Amazon EMR. Managed policies offer the benefit of updating automatically if permission requirements change. These policies not only include actions for Amazon EMR; they also include actions for Amazon EC2, Amazon S3, and Amazon CloudWatch, which Amazon EMR uses to perform actions like launching instances, writing log files, and managing Hadoop jobs and tasks. If you

need to create custom policies, it is recommended that you begin with the managed policies and edit them according to your requirements.

For information about how to attach policies to IAM users (principals), see [Working with Managed Policies Using the AWS Management Console](#) in the *IAM User Guide*.

IAM Managed Policy for Full Access

To grant all the required actions for Amazon EMR, attach the **AmazonElasticMapReduceFullAccess** *managed policy*. The content of this policy statement is shown below. It reveals all the actions that Amazon EMR requires for other services.

Note

Because the **AmazonElasticMapReduceFullAccess** policy is automatically updated, the policy shown here may be out-of-date. Use the AWS Management Console to view the current policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:*",
        "cloudformation:CreateStack",
        "cloudformation:DescribeStackEvents",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:AuthorizeSecurityGroupEgress",
        "ec2:CancelSpotInstanceRequests",
        "ec2:CreateRoute",
        "ec2:CreateSecurityGroup",
        "ec2:CreateTags",
        "ec2>DeleteRoute",
        "ec2>DeleteTags",
        "ec2>DeleteSecurityGroup",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeInstances",
        "ec2:DescribeKeyPairs",
        "ec2:DescribeRouteTables",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSpotInstanceRequests",
        "ec2:DescribeSpotPriceHistory",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeVpcs",
        "ec2:DescribeRouteTables",
        "ec2:DescribeNetworkAcls",
        "ec2:CreateVpcEndpoint",
        "ec2:ModifyImageAttribute",
        "ec2:ModifyInstanceAttribute",
        "ec2:RequestSpotInstances",
        "ec2:RevokeSecurityGroupEgress",
        "ec2:RunInstances",
        "ec2:TerminateInstances",
        "elasticmapreduce:*",
        "iam:GetPolicy",
        "iam:GetPolicyVersion",
        "iam:ListRoles",
        "iam:PassRole",
        "kms:List*",
        "s3:*",
        "sdb:*",
        "support:CreateCase",
        "support:DescribeServices",

```

```
        "support:DescribeSeverityLevels"
      ],
      "Resource": "*"
    }
  ]
}
```

Note

The `ec2:TerminateInstances` action enables the IAM user to terminate any of the Amazon EC2 instances associated with the IAM account, even those that are not part of an EMR cluster.

IAM Managed Policy for Read-Only Access

To grant read-only privileges to Amazon EMR, attach the **AmazonElasticMapReduceReadOnlyAccess** managed policy. The content of this policy statement is shown below. Wildcard characters for the `elasticmapreduce` element specify that only actions that begin with the specified strings are allowed. Keep in mind that because this policy does not explicitly deny actions, a different policy statement may still be used to grant access to specified actions.

Note

Because the **AmazonElasticMapReduceReadOnlyAccess** policy is automatically updated, the policy shown here may be out-of-date. Use the AWS Management Console to view the current policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:Describe*",
        "elasticmapreduce:List*",
        "elasticmapreduce:ViewEventsFromAllClustersInConsole",
        "s3:GetObject",
        "s3:ListAllMyBuckets",
        "s3:ListBucket",
        "sdb:Select",
        "cloudwatch:GetMetricStatistics"
      ],
      "Resource": "*"
    }
  ]
}
```

Use Cluster Tags for Fine-Grained Access Control

You can use the `Condition` element (also called a `Condition` block) along with the following Amazon EMR condition context keys in an IAM user-based policy to control access based on cluster tags:

- Use the `elasticmapreduce:ResourceTag/TagKeyString` condition context key to allow or deny user actions on clusters with specific tags.
- Use the `elasticmapreduce:RequestTag/TagKeyString` condition context key to require a specific tag with actions/API calls.

Important

The condition context keys apply only to those Amazon EMR API actions that accept `ClusterID` as a request parameter. Because the [ModifyInstanceGroups](#) action does not accept `ClusterID` as an input, you can neither allow nor deny permissions for this action based on cluster tags. This is important to consider when you plan your authorization strategy.

For a complete list of Amazon EMR actions, see the API action names in the [Amazon EMR API Reference](#). For more information about the `Condition` element and condition operators, see [IAM Policy Elements Reference](#) in the *IAM User Guide*, particularly [String Condition Operators](#). For more information about adding tags to EMR clusters, see [Tagging Amazon EMR Clusters](#).

Example Amazon EMR Policy Statements

The following examples demonstrate different scenarios and ways to use condition operators with Amazon EMR condition context keys. These IAM policy statements are intended for demonstration purposes only and should not be used in production environments. There are multiple ways to combine policy statements to grant and deny permissions according to your requirements. For more information about planning and testing IAM policies, see the [IAM User Guide](#).

Allow Actions Only on Clusters with Specific Tag Values

The examples below demonstrate a policy that allows a user to perform actions based on the cluster tag `department` with the value `dev` and also allows a user to tag clusters with that same tag. The final policy example demonstrates how to deny privileges to tag EMR clusters with anything but that same tag.

Important

Explicitly denying permission for tagging actions is an important consideration. This prevents users from granting permissions to themselves through cluster tags that you did not intend to grant. If the actions shown in the last example had not been denied, a user could add and remove tags of their choosing to any cluster, and circumvent the intention of the preceding policies.

In the following policy example, the `StringEquals` condition operator tries to match `dev` with the value for the tag `department`. If the tag `department` hasn't been added to the cluster, or doesn't contain the value `dev`, the policy doesn't apply, and the actions aren't allowed by this policy. If no other policy statements allow the actions, the user can only work with clusters that have this tag with this value.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt14793345241244",
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:DescribeCluster",
        "elasticmapreduce:ListSteps",
        "elasticmapreduce:TerminateJobFlows ",
        "elasticmapreduce:SetTerminationProtection ",
        "elasticmapreduce:ListInstances",
        "elasticmapreduce:ListInstanceGroups",
        "elasticmapreduce:ListBootstrapActions",
        "elasticmapreduce:DescribeStep"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "elasticmapreduce:ResourceTag/department": "dev"
        }
      }
    }
  ]
}
```

You can also specify multiple tag values using a condition operator. For example, to allow all actions on clusters where the `department` tag contains the value `dev` or `test`, you could replace the condition block in the earlier example with the following.

```
    "Condition": {
      "StringEquals": {
        "elasticmapreduce:ResourceTag/department": ["dev", "test"]
      }
    }
  }
```

As in the preceding example, the following example policy looks for the same matching tag: the value `dev` for the `department` tag. In this case, however, the `RequestTag` condition context key specifies that the policy applies during tag creation, so the user must create a tag that matches the specified value.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1479334524000",
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:RunJobFlow",
        "iam:PassRole"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "elasticmapreduce:RequestTag/department": "dev"
        }
      }
    }
  ]
}
```

In the following example, the EMR actions that allow the addition and removal of tags is combined with a `StringNotEquals` operator specifying the `dev` tag we've seen in earlier examples. The effect of this policy is to deny a user the permission to add or remove any tags on EMR clusters that are tagged with a `department` tag that contains the `dev` value.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "elasticmapreduce:AddTags",
        "elasticmapreduce:RemoveTags"
      ],
      "Condition": {
        "StringNotEquals": {
          "elasticmapreduce:ResourceTag/department": "dev"
        }
      },
      "Resource": [
        "*"
      ]
    }
  ]
}
```

```
}  
]  
}
```

Allow Actions on Clusters with a Specific Tag, Regardless of Tag Value

You can also allow actions only on clusters that have a particular tag, regardless of the tag value. To do this, you can use the `Null` operator. For more information, see [Condition Operator to Check Existence of Condition Keys](#) in the *IAM User Guide*. For example, to allow actions only on EMR clusters that have the `department` tag, regardless of the value it contains, you could replace the Condition blocks in the earlier example with the following one. The `Null` operator looks for the presence of the tag `department` on an EMR cluster. If the tag exists, the `Null` statement evaluates to false, matching the condition specified in this policy statement, and the appropriate actions are allowed.

```
"Condition": {  
  "Null": {  
    "elasticmapreduce:ResourceTag/department": "false"  
  }  
}
```

The following policy statement allows a user to create an EMR cluster only if the cluster will have a `department` tag, which can contain any value.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Action": [  
        "elasticmapreduce:RunJobFlow",  
        "iam:PassRole"  
      ],  
      "Condition": {  
        "Null": {  
          "elasticmapreduce:RequestTag/department": "false"  
        }  
      },  
      "Effect": "Allow",  
      "Resource": [  
        "*"   
      ]  
    }  
  ]  
}
```

Require Users to Add Tags When Creating a Cluster

The following policy statement allows a user to create an EMR cluster only if the cluster will have a `department` tag that contains the value `dev` when it is created.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Action": [  
        "elasticmapreduce:RunJobFlow",  

```

```
        "iam:PassRole"
      ],
      "Condition": {
        "StringEquals": {
          "elasticmapreduce:RequestTag/department": "dev"
        }
      },
      "Effect": "Allow",
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Configure IAM Roles for Amazon EMR and Applications

Amazon EMR and applications such as Hadoop need permission to access other AWS resources when running jobs on behalf of users. Two IAM roles, a *service role* and an Amazon EC2 *instance profile*, are required to grant those permissions. In most cases, default policies are adequate, but you can modify the policies to tailor access to specific requirements. A user must specify the service role and Amazon EC2 instance profile in the cluster definition when a cluster is created. The permissions determine which AWS resources a service can access, and what the service is allowed to do with those resources. The permissions granted to the service role and Amazon EC2 instance profile are separate from the permissions granted to the IAM user so that an AWS administrator can manage them separately and tailor a permissions policy that closely fits the usage patterns of the cluster.

The service role defines the allowable actions for Amazon EMR based on granted permissions. When the user accesses the cluster, Amazon EMR assumes this IAM role, gets the permissions of the assumed role, and then tries to execute requests with those permissions. A similar process occurs for applications using the Amazon EC2 instance profile, which determines permissions for applications that run on EC2 instances. For example, when Hive, an application on the cluster, needs to write output to an Amazon S3 bucket, the Amazon EC2 instance profile determines whether Hive has permissions to do that.

An Amazon EMR service role and an Amazon EC2 instance profile are required for all clusters in all regions. For more information about service and Amazon EC2 roles, see [Use Cases: Roles for Users, Applications, and Services](#) and [Use roles for applications that run on Amazon EC2 instances](#) in the *IAM User Guide*.

Note

The user who sets up the roles for use with Amazon EMR should be an IAM user with administrative permissions. We recommend that all administrators use AWS MFA (multi-factor authentication).

Topics

- [Default IAM Roles for Amazon EMR \(p. 106\)](#)
- [Create and Use IAM Roles for Amazon EMR \(p. 107\)](#)
- [Launch a Cluster with IAM Roles \(p. 111\)](#)
- [Use IAM Roles with Applications That Call AWS Services Directly \(p. 112\)](#)

Default IAM Roles for Amazon EMR

To simplify using IAM roles, Amazon EMR has two predefined roles, and two default managed policies, which you can attach by default when creating a cluster:

- For the Amazon EMR service role, the **EMR_DefaultRole** role is attached to the **AmazonElasticMapReduceRole** managed policy.
- For the Amazon EMR instance profile for Amazon EC2, the **EMR_EC2_DefaultRole** role is attached to the **AmazonElasticMapReduceforEC2Role** managed policy.

Note

In addition to these default roles, if you use automatic scaling with Amazon EMR, the feature must have permissions to add and terminate clusters on your behalf. A default role, **EMR_AutoScaling_DefaultRole**, which is configured with the appropriate role policy and trust policy, is available for this purpose. Automatic scaling uses this IAM role to scale nodes on your behalf. You must add this role to an EMR cluster using `--auto-scaling-role EMR_AutoScaling_DefaultRole`. It is not added when you use `--use-default-roles`. For more information, see [Using Automatic Scaling in Amazon EMR \(p. 191\)](#).

AWS managed policies are policies that are created and managed by AWS to attach to roles required by services. For more information, see [Managed Policies and Inline Policies](#) in the *IAM User Guide*. You can view the most up-to-date managed policies for the Amazon EMR service role and EC2 role in the **Policies** tab in the IAM console.

Create and Use IAM Roles for Amazon EMR

There are three ways to create IAM roles:

- Use the Amazon EMR console to create the default roles. For more information, see [Create and Use IAM Roles with the Amazon EMR Console \(p. 108\)](#).
- Use the AWS CLI to create the default roles using the `create-default-roles` subcommand. For more information, see [Create and Use IAM Roles with the AWS CLI \(p. 108\)](#).
- Use the IAM console or API to create roles. For more information, see [Creating a Role for an AWS Service](#) in the *IAM User Guide*.

Note

When you use the IAM console to create a role where Amazon EC2 is the principal, IAM automatically creates an instance profile with the same name as the role, to contain the role and pass information to EC2 instances. For more information, see [Instance Profiles](#) in the *IAM User Guide*.

If you are using an IAM user and creating default roles for a cluster, your IAM user must have the following permissions:

- `iam:CreateRole`
- `iam:PutRolePolicy`
- `iam:CreateInstanceProfile`
- `iam:AddRoleToInstanceProfile`
- `iam:ListRoles`
- `iam:GetPolicy`
- `iam:GetInstanceProfile`
- `iam:GetPolicyVersion`
- `iam:AttachRolePolicy`
- `iam:PassRole`

The `iam:PassRole` permission allows cluster creation. The remaining permissions allow creation of the default roles.

Create and Use IAM Roles with the Amazon EMR Console

AWS customers whose accounts were created after release of Amazon EMR roles are required to specify an Amazon EMR (service) role and an EC2 instance profile in all regions when using the console. You can create default roles at cluster launch using the console, or you can specify other roles you may already be using.

Note

The `EMR_AutoScaling_DefaultRole` cannot be created using the console.

To create and use IAM roles with the console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create Cluster**.
3. In the **Security and Access** section, in the **IAM Roles** subsection, for **Roles configuration**, choose **Default**. If the default roles do not exist, they are created for you (assuming that you have appropriate permissions). If the roles exist, they are used for your cluster. After the roles are created, they are visible in the IAM console.

Note

To use custom roles with your cluster, choose **Custom** and select the existing roles from the list.

Create and Use IAM Roles with the AWS CLI

You can create the default Amazon EMR (service) role and EC2 instance profile using the CLI. After the roles are created, they are visible in the IAM console. If not already present, the roles are also auto-populated in the AWS CLI configuration file located at `~/.aws/config` on Unix, Linux, and OS X systems or at `C:\Users\USERNAME\.aws\config` on Windows systems. After creation, you can use the default roles when you launch a cluster.

To create and use IAM roles with the AWS CLI

To create default roles using the AWS CLI, type the `create-default-roles` subcommand. To use the default roles at cluster launch, type the `create-cluster` subcommand with the `--use-default-roles` parameter. This command does not create the `EMR_AutoScaling_DefaultRole`.

1. Type the following command to create default roles using the AWS CLI:

```
aws emr create-default-roles
```

The output of the command lists the contents of the default Amazon EMR role, `EMR_DefaultRole`; the the default EC2 instance profile, `EMR_EC2_DefaultRole`. The AWS CLI configuration file is populated with these role names for the `service_role` and `instance_profile` values. For example, after this command, the configuration file might look like the following:

```
[default]
output = json
region = us-east-1
aws_access_key_id = myAccessKeyID
aws_secret_access_key = mySecretAccessKey
emr =
    service_role = EMR_DefaultRole
    instance_profile = EMR_EC2_DefaultRole
```

You can also modify this file to use your own custom role and the AWS CLI uses that role by default.

2. If the default roles already exist, you can use them when launching a cluster. Type the following command to use existing default roles when launching a cluster and replace *myKey* with the name of your EC2 key pair.

Important

This command will not add the `EMR_AutoScaling_DefaultRole`. You must explicitly add this role using the `--auto-scaling-role EMR_AutoScaling_DefaultRole` option with the `create-cluster` command. For more information, see [Creating the IAM Role for Automatic Scaling](#) (p. 191).

```
aws emr create-cluster --name "Test cluster" --release-label emr-4.0.0 --applications
Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --instance-
type m3.xlarge --instance-count 3
```

Attach Default IAM Roles to Managed Policies in the IAM Console

If you need to customize the default IAM roles, we recommend that you begin by creating the default roles, and then modify those roles as needed. In addition, if you created default roles before 6/11/2015, Amazon EMR did not attach managed policies to the role by default.

Use the following procedures to attach the IAM roles for Amazon EMR to their respective default managed policies. You can then view the default managed policies and edit them for customization as required.

Attach the Default Service Role to a Managed Policy for Amazon EMR

The `EMR_DefaultRole` consists of a role policy and a trust policy. You can view the most up-to-date **AmazonElasticMapReduceRole** in the **Policies** tab in the [IAM console](#).

To attach `EMR_DefaultRole` to the **AmazonElasticMapReduceRole** managed policy using the console

Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.

1. Choose **Policies, AmazonElasticMapReduceRole**.
2. Under **Attached Entities**, choose **Attach, EMR_DefaultRole, Attach**.

To attach `EMR_DefaultRole` to the **AmazonElasticMapReduceRole** managed policy using the AWS CLI

- Use the following syntax to attach your pre-existing default role to the managed policy:

```
$ aws iam attach-role-policy --role-name EMR_DefaultRole \
--policy-arn arn:aws:iam::aws:policy/service-role/AmazonElasticMapReduceRole
$ aws iam delete-role-policy --role-name EMR_DefaultRole --policy-name EMR_DefaultRole
```

Attach the Default EC2 Instance Profile to a Managed Policy for Amazon EMR

The default EC2 role consists of a role policy and a trust policy. You can view the most up-to-date **AmazonElasticMapReduceforEC2Role** in the **Policies** tab in the [IAM console](#).

To attach `EMR_EC2_DefaultRole` to the managed policy using the console

Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.

1. Choose **Policies, AmazonElasticMapReduceforEC2Role**.
2. Under **Attached Entities**, choose **Attach, EMR_EC2_DefaultRole**, and **Attach**.

To attach EMR_EC2_DefaultRole to the managed policy using the CLI

- Use the following AWS CLI syntax to attach your pre-existing default role to the managed policy:

```
$ aws iam attach-role-policy --role-name EMR_EC2_DefaultRole\  
--policy-arn arn:aws:iam::aws:policy/service-role/AmazonElasticMapReduceforEC2Role  
$ aws iam delete-role-policy --role-name EMR_EC2_DefaultRole --policy-name  
EMR_EC2_DefaultRole
```

Custom IAM Roles for Amazon EMR

If the default IAM roles provided by Amazon EMR do not meet your needs, you can create custom roles instead. For example, if your application does not access Amazon DynamoDB, you can leave out DynamoDB permissions in your custom IAM role.

For more information about creating and managing IAM roles, see the following topics in the *IAM User Guide*:

- [Creating a Role](#)
- [Modifying a Role](#)
- [Deleting a Role](#)

We recommend that you begin with the permissions in the managed policies

AmazonElasticMapReduceforEC2Role and **AmazonElasticMapReduceRole** when developing custom IAM roles to use with Amazon EMR. You can then copy the contents of these default roles, create new IAM roles, paste in the copied permissions, and modify the pasted permissions.

The following is an example of a custom Amazon EC2 instance profile for use with Amazon EMR. This example is for a cluster that does not use Amazon RDS, or DynamoDB.

The access to Amazon SimpleDB is included to permit debugging from the console although in releases greater than 4.1, `sqs:*` is required for this purpose. Access to CloudWatch is included so the cluster can report metrics. Amazon SNS and Amazon SQS permissions are included for messaging. The minimum Amazon SQS permissions required for Amazon EMR are: `sqs:SendMessage` and `sqs:QueueExists`.

```
{  
  "Statement": [  
    {  
      "Action": [  
        "cloudwatch:*",  
        "ec2:Describe*",  
        "elasticmapreduce:Describe*",  
        "s3:*",  
        "sdb:*",  
        "sns:*",  
        "sqs:*"  
      ],  
      "Effect": "Allow",  
      "Resource": "*"   
    }  
  ]  
}
```

Important

The IAM role name and the instance profile name must match exactly when you use either the Amazon EMR console or CLI. The console shows IAM role names in the **EC2 instance profile** list. The CLI interprets the name as instance profile names. For this reason, if you use the IAM CLI or API to create an IAM role and its associated instance profile, we recommend that you give the new IAM role the same name as its associated instance profile. By default, if you create an IAM role with the IAM console and do not specify an instance profile name, the instance profile name is the same as the IAM role name.

In some situations, you might need to work with an IAM role whose associated instance profile does not have the same name as the role. This can occur if you use AWS CloudFormation to manage IAM roles for you, because AWS CloudFormation adds a suffix to the role name to create the instance profile name. In this case, you can use the Amazon EMR API or CLI to specify the instance profile name. Unlike the console, the API and CLI do not interpret an instance profile name as IAM role name. In the API, you can call the `RunJobFlow` action and pass the instance profile name for the `JobFlowRole` parameter. In the CLI, you can specify the instance profile name for the `--ec2-attributes InstanceProfile` option of `aws emr create-cluster` command.

Launch a Cluster with IAM Roles

The IAM user creating clusters needs permissions to retrieve and assign roles to Amazon EMR and EC2 instances. If the IAM user lacks these permissions, you get the error **User account is not authorized to call EC2**. You assign the correct role by creating the default roles as discussed in [Default IAM Roles for Amazon EMR \(p. 106\)](#).

To launch a cluster with IAM roles using the console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**.
3. For **IAM Roles**, choose **Default** or **Custom**.
 - a. If you choose **Default**, you can optionally click **View policies for default roles**.
 - b. If you choose **Custom**, specify the IAM roles using the **EMR role** and **EC2 instance profile** fields. For more information, see [Create and Use IAM Roles with the Amazon EMR Console \(p. 108\)](#)

To launch a cluster with IAM roles using the AWS CLI

You can specify an Amazon EMR service role and EC2 instance profile using the AWS CLI. When launching a cluster, type the `create-cluster` subcommand with the `--service-role` and `--ec2-attributes InstanceProfile` parameters.

- Type the following command to specify an Amazon EMR role and EC2 instance profile when launching a cluster. This example uses the default Amazon EMR role, `EMR_DefaultRole`, and the default EC2 instance profile, `EMR_EC2_DefaultRole`.

```
aws emr create-cluster --name "Test cluster" --release-label emr-4.1.0 --
applications Name=Hive Name=Pig --service-role EMR_DefaultRole --ec2-attributes
InstanceProfile=EMR_EC2_DefaultRole,KeyName=myKey --instance-type m3.xlarge --
instance-count 3
```

Alternatively, you can use the `--use-default-roles` option, which assumes those roles have been created:

```
aws emr create-cluster --name "Test cluster" --release-label emr-4.1.0 --applications
Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --instance-
type m3.xlarge --instance-count 3
```

Another alternative is to set the service role and the instance profile located in the AWS CLI configuration file. For more information, see [Create and Use IAM Roles for Amazon EMR \(p. 107\)](#)

Note

When you specify the instance count without using the `--instance-groups` parameter, a single master node is launched, and the remaining instances are launched as core nodes. All nodes use the instance type specified in the command.

For more information, see [Amazon EMR commands in the AWS CLI](#).

Use IAM Roles with Applications That Call AWS Services Directly

Applications running on the EC2 instances of a cluster can use the instance profile to obtain temporary security credentials when calling AWS services.

The version of Hadoop available on AMI 2.3.0 and later has already been updated to make use of IAM roles. If your application runs strictly on top of the Hadoop architecture, and does not directly call any service in AWS, it should work with IAM roles with no modification.

If your application calls services in AWS directly, you need to update it to take advantage of IAM roles. This means that instead of obtaining account credentials from `/etc/hadoop/conf/core-site.xml` on the EC2 instances in the cluster, your application now uses an SDK to access the resources using IAM roles, or calls the EC2 instance metadata to obtain the temporary credentials.

To access AWS resources with IAM roles using an SDK

- The following topics show how to use several of the AWS SDKs to access temporary credentials using IAM roles. Each topic starts with a version of an application that does not use IAM roles and then walks you through the process of converting that application to use IAM roles.
 - [Using IAM Roles for Amazon EC2 Instances with the SDK for Java](#) in the *AWS SDK for Java Developer Guide*
 - [Using IAM Roles for Amazon EC2 Instances with the SDK for .NET](#) in the *AWS SDK for .NET Developer Guide*
 - [Using IAM Roles for Amazon EC2 Instances with the SDK for PHP](#) in the *AWS SDK for PHP Developer Guide*
 - [Using IAM Roles for Amazon EC2 Instances with the SDK for Ruby](#) in the *AWS SDK for Ruby Developer Guide*

To obtain temporary credentials from EC2 instance metadata

- Call the following URL from an EC2 instance that is running with the specified IAM role, which returns the associated temporary security credentials (AccessKeyId, SecretAccessKey, SessionToken, and Expiration). The example that follows uses the default instance profile for Amazon EMR, `EMR_EC2_DefaultRole`.

```
GET http://169.254.169.254/latest/meta-data/iam/security-credentials/EMR_EC2_DefaultRole
```

For more information about writing applications that use IAM roles, see [Granting Applications that Run on Amazon EC2 Instances Access to AWS Resources](#).

For more information about temporary security credentials, see [Using Temporary Security Credentials](#) in the *Using Temporary Security Credentials* guide.

Configure Security Groups

A security group acts as a virtual firewall for your EC2 instances to control inbound and outbound traffic. For each security group, one set of rules controls the inbound traffic to instances, and a separate set of rules controls outbound traffic. In Amazon EMR, there are two types of security groups for your EC2 instances:

- Amazon EMR–managed security groups for the master and core/task instances: you can choose the default master and core/task security groups (ElasticMapReduce-master, ElasticMapReduce-slave, ElasticMapReduce-Master-Private, ElasticMapReduce-Slave-Private, and ElasticMapReduce-ServiceAccess), or you can specify your own security groups for the master and core/task instances
- Additional security groups: you can specify security groups that apply to your Amazon EC2 instances in addition to the Amazon EMR–managed security groups

Security groups need to be specified at cluster launch using the console, CLI, API, or SDK. Instances in running clusters cannot be assigned new security groups, but you can edit or add rules to existing security groups. Edited rules and new rules added to security groups take effect when the rules are saved.

For more information, see [Amazon EC2 Security Groups](#) in the *Amazon EC2 User Guide for Linux Instances*. For more information about Amazon VPC security groups, see [Security Groups for Your VPC](#) in the *Amazon VPC User Guide*.

Topics

- [Amazon EMR–Managed Security Groups](#) (p. 113)
- [Amazon EMR Additional Security Groups](#) (p. 117)

Amazon EMR–Managed Security Groups

When you launch an Amazon EMR cluster, you must specify one Amazon EMR–managed security group for the master instance, one for Amazon EMR–managed security group for the core/task (slave) instances, and optionally, one for the Amazon EMR resources used to manage clusters in private subnets. The core/task instances share the same security group.

You can use the default Amazon EMR–managed security groups for cluster resources or you can choose your own security groups for the master and core/task instances. You cannot combine your own security groups with the default groups.

When you launch a cluster using the console, if the default security groups do not exist, the Amazon EMR–managed security groups fields are populated with `Create ElasticMapReduce-master` and `Create ElasticMapReduce-slave` or `Create ElasticMapReduce-Master-Private`, `Create ElasticMapReduce-Slave-Private`, and `Create ElasticMapReduce-ServiceAccess`. If the default security groups exist, the fields are populated with the default security group IDs; for example, `Default: sg-01XXXX6a` (`ElasticMapReduce-master`) and `Default: sg-07XXXX6c` (`ElasticMapReduce-slave`). To use your own security groups, choose them from the lists. For more information about the default Amazon EMR–managed security group options, see [Default Options for Amazon EMR–Managed Security Groups](#) (p. 115).

Amazon EMR-specific inbound and outbound access rules are written to and maintained in the Amazon EMR–managed security groups. Whether you choose the default security groups or your own, Amazon EMR modifies the security group rules to ensure proper communication between instances in a cluster.

For more information about the rules written to the Amazon EMR–managed security groups, see [Rules in Amazon EMR–Managed Security Groups \(p. 116\)](#).

If you need to control instance membership in the Amazon EMR–managed security groups, you can create your own security groups for the master and core/task instances. For more information, see [Security Groups for Your VPC](#) in the *Amazon VPC User Guide*. When you create your own security groups, any inbound and outbound rules required for proper intra-cluster communication are written to the groups with some exceptions.

Typically, you would create your own Amazon EMR–managed security groups to isolate clusters so they cannot communicate with each other. This alleviates the need to create separate VPCs or AWS accounts for each of your clusters. For example, to isolate Amazon EMR clusters in a VPC, you create a security group for each cluster's master node, and you create a security group for each cluster's core/task nodes. When you launch a cluster, the rules in your security groups are applied only to the instances in that cluster, effectively preventing instances in separate clusters from communicating with each other.

If you need separate clusters to communicate, and you are using your own security groups, you can create an additional security group containing the rules necessary for the cluster instances to communicate with each other, or you can use the same security groups for both clusters. If you do not specify an additional security group when launching the clusters, and the clusters use different security groups, the clusters cannot communicate with each other unless you modify the rule sets in your master and core/task security groups. For more information about additional security groups, see [Amazon EMR Additional Security Groups \(p. 117\)](#).

To specify Amazon EMR–managed security groups using the console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**.
3. In the **Security and Access** section, in the **EC2 Security Groups** subsection:
 - For **Master**, if the default security groups do not exist, the field is populated with **Create ElasticMapReduce-master** Or **Create ElasticMapReduce-Master-Private**. If the default security groups exist, the field is populated with the default security group ID; for example, **Default: sg-01XXXX6a (ElasticMapReduce-master)**. To specify your own master security group, choose it from the list.
 - For **Core & Task**, if the default security groups do not exist, the field is populated with **Create ElasticMapReduce-slave** Or **Create ElasticMapReduce-Slave-Private**. If the default security groups exist, the field is populated with the default security group ID; for example, **Default: sg-07XXXX6c (ElasticMapReduce-slave)**. To specify your own core/task security group, choose it from the list.
 - (Optional) For **Service Access** to clusters in private subnets, if the default security groups do not exist, the field is populated with **Create ElasticMapReduce-ServiceAccess**. If the default security groups exist, the field is populated with the default security group ID; for example, **Default: sg-4dXXXX34 (ElasticMapReduce-ServiceAccess)**. To specify your own service access security group, choose it from the list.
4. Proceed with creating the cluster as described in [Plan and Configure Clusters \(p. 24\)](#).

To specify Amazon EMR–managed security groups using the AWS CLI

Use the `create-cluster` command with the `--emr-managed-master-security-group` and `--emr-managed-slave-security-group` parameters.

If you are using the default options for Amazon EMR–managed security groups, no additional parameters are required. Use the `create-cluster` command as you normally would using the CLI. If the default security groups do not exist, they are created before your cluster is launched. If they do exist, they are automatically assigned.

Note

Amazon EMR-managed security groups are not supported in the Amazon EMR CLI.

1. To launch a cluster using the default security group options, type the following command, replace *myKey* with the name of your Amazon EC2 key pair.

```
aws emr create-cluster --name "Test cluster" --release-label emr-4.2.0 --applications
Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --instance-
type m3.xlarge --instance-count 3
```

When you specify the instance count without using the `--instance-groups` parameter, a single master node is launched, and the remaining instances are launched as core nodes. All nodes use the instance type specified in the command.

2. To launch a cluster using your own security groups, type the following command, replace *myKey* with the name of your Amazon EC2 key pair, replace *masterSecurityGroupId* with the ID of the master security group, and replace *slaveSecurityGroupId* with the ID of the core/task security group.

```
aws emr create-cluster --name "Test cluster" --release-label
emr-4.2.0 --applications Name=Hive Name=Pig --ec2-attributes
KeyName=myKey,EmrManagedMasterSecurityGroup=sg-
masterId,EmrManagedSlaveSecurityGroup=sg-slaveId --instance-type m3.xlarge --instance-
count 3 --use-default-roles
```

Note

For private subnets, you can additionally specify

`ServiceAccessSecurityGroup=sg-service-accessId` with the `--ec2-attributes` parameter.

When you specify the instance count without using the `--instance-groups` parameter, a single master node is launched, and the remaining instances are launched as core nodes. All nodes use the instance type specified in the command.

3. To retrieve security group information for your cluster, type the following command and replace *j-1K48XXXXXXHCB* with your cluster ID:

```
aws emr describe-cluster --cluster-id j-1K48XXXXXXHCB
```

For more information, see [Amazon EMR commands in the AWS CLI](#).

Default Options for Amazon EMR-Managed Security Groups

If you launch an Amazon EMR cluster using the default security groups, two groups are created for public subnets: ElasticMapReduce-master and ElasticMapReduce-slave. For private subnets, three groups are created:

- Create ElasticMapReduce-Master-Private
- Create ElasticMapReduce-Slave-Private
- Create ElasticMapReduce-ServiceAccess

The inbound and outbound access rules written to these groups ensure that the master and core/task instances in a cluster can communicate properly.

In addition, if you launch other Amazon EMR clusters in the same VPC using the default security groups, the instances in those clusters can communicate with the instances in any other Amazon EMR cluster within that VPC whose instances also belong to the same security groups.

You can launch a cluster with the default security groups using the console, the API, the CLI, or the SDK. If you use the default security groups, there is no need to change your existing code or to add parameters to your CLI commands.

You can launch a cluster with the default security groups using the console. If the default security groups do not exist, they are created before your cluster is launched. If they do exist, they are automatically assigned.

Rules in Amazon EMR–Managed Security Groups

The tables in this section list the inbound and outbound access rules added to the Amazon EMR–managed security groups. IP address ranges and rules are automatically updated for Amazon EMR–managed security groups.

Warning

We do not recommend creating an inbound rule for any protocol or port that allows inbound traffic from all IP addresses (0.0.0.0/0). This opens access to everyone and creates a security vulnerability.

The following rules are added to Amazon EMR–managed master security groups:

Type	Protocol	Port Range	Source	Details
<i>Inbound rules</i>				
All ICMP	ICMP	All	The default ElasticMapReduce-master security group ID or the master security group ID that you specify; for example, sg-88XXXXed	Allows inbound traffic from any instance in the ElasticMapReduce-master security group. By default, the master nodes in all Amazon EMR clusters in a single VPC can communicate with each other over any TCP, UDP, or ICMP port. If you choose your own master security group, only master instances in the group can communicate with each other over any TCP, UDP, or ICMP port.
All TCP	TCP	All		
All UDP	UDP	All		
All ICMP	ICMP	All	The default ElasticMapReduce-slave security group ID or the core/task security group ID that you specify; for example, sg-8bXXXXee	Allows inbound traffic from any instance in the ElasticMapReduce-slave security group. By default, the master node accepts inbound communication from any core/task node in any Amazon EMR cluster in a single VPC over any TCP, UDP, or ICMP port. If you choose your own core/task security group, only core/task instances in this group can communicate with the master node over any TCP, UDP, or ICMP port.
All TCP	TCP	All		
All UDP	UDP	All		
HTTPS	TCP	8443	Various Amazon IP address ranges	Allows the cluster manager to communicate with the master nodes in each Amazon EMR cluster in a single VPC.
<i>Outbound rules</i>				
All traffic	All	All	0.0.0.0/0	Provides outbound access to the Internet from any instance in the ElasticMapReduce-master security group or the group that you specify.

The following rules are added to Amazon EMR–managed core/task security groups:

Type	Protocol	Port Range	Source	Details
<i>Inbound rules</i>				
All ICMP	ICMP	All	The default ElasticMapReduce-master security group ID or the master security group ID that you specify; for example, sg-88XXXXed	Allows inbound traffic from any instance in the ElasticMapReduce-master security group or the group that you specify. By default, the core/task nodes in all Amazon EMR clusters in a single VPC accept inbound communication from master nodes over any TCP, UDP, or ICMP port. If you choose your own master security group, only master instances in this group can communicate with the core/task nodes over any TCP, UDP, or ICMP port.
All TCP	TCP	All		
All UDP	UDP	All		
All ICMP	ICMP	All	The default ElasticMapReduce-slave security group ID or the core/task security group ID that you specify; for example, sg-8bXXXXee	Allows inbound traffic from any instance in the ElasticMapReduce-slave security group. By default, the core/task nodes accept inbound communication from any other core/task node in any Amazon EMR cluster in a single VPC over any TCP, UDP, or ICMP port. If you choose your own core/task security group, only core/task instances in this group can communicate with each other over any TCP, UDP, or ICMP port.
All TCP	TCP	All		
All UDP	UDP	All		
Custom TCP	TCP	8443		
<i>Outbound rules</i>				
All traffic	All	All	0.0.0.0/0	Provides outbound access to the Internet from all instances in the ElasticMapReduce-slave security group or the group that you specify.

Amazon EMR Additional Security Groups

Whether you use the default managed security groups or your own custom managed security groups, you can assign additional security groups to the master and core/task instances in your cluster. Applying additional security groups gives you the ability to apply additional rules to your security groups so they do not have to be modified. Additional security groups are optional. They can be applied to the master group, core/task group, both groups, or neither group. You can also apply the same additional security groups to multiple clusters.

For example, if you are using your own managed security groups, and you want to allow inbound SSH access to the master group for a particular cluster, you can create an additional security group containing the rule and add it to the master security group for that cluster. Additional security groups are not modified by or maintained by Amazon EMR.

Typically, additional security groups are used to:

- Add access rules to instances in your cluster that are not present in the Amazon EMR–managed security groups

- Give particular clusters access to a specific resource such as a Amazon Redshift database

Security groups are restrictive by default. They reject all traffic. You can add a rule to allow traffic on a particular port to your custom or additional security groups. If there is more than one rule for a specific port in two security groups that apply to the same instances, the most permissive rule is applied. For example, if you have a rule that allows SSH access via TCP port 22 from IP address 203.0.113.1 and another rule that allows access to TCP port 22 from any IP address (0.0.0.0/0), the rule allowing access by any IP address takes precedence.

You can apply up to four additional security groups to both the master and core/task security groups. The number of additional groups allowed is dependent upon:

- The number of security groups allowed for your account
- The number of individual rules allowed for your account

For more information about rule limits in VPC security groups, see [Security Groups for Your VPC](#) in the *Amazon VPC User Guide*. You can assign the same additional security groups to both the master and core/task security groups.

Additional security groups can be applied using the console, the API, the CLI, or the SDK.

To specify additional security groups using the console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**.
3. Choose **Go to advanced options**.
4. In the **Security and Access** section, in the **EC2 Security Groups** subsection:
 - For **Master**, choose either the default security group or a custom security group from the list.
 - In the **Additional security groups** column, choose the icon to add up to four additional groups to the master security group.
 - For **Core & Task**, choose either the default security group or a custom security group from the list.
 - In the **Additional security groups** column, choose the icon to add up to four additional groups to the core/task security group.

Note

You cannot combine custom and default security groups.

5. Proceed with creating the cluster as described in [Plan and Configure Clusters \(p. 24\)](#).

To specify additional security groups using the AWS CLI

Use the `create-cluster` command with the `--ec2-attributes` parameter, specifying security group IDs for the `AdditionalSlaveSecurityGroups` and `AdditionalMasterSecurityGroups` variables. Additional security groups are optional. They can be applied to the master group, core/task group, both groups, or neither group.

Note

Amazon EMR-managed security groups and additional security groups are not supported in the Amazon EMR CLI.

1. To launch a cluster using additional security groups, type the following command. Replace *myKey* with the name of your Amazon EC2 key pair, and replace *securityGroupId* with the ID of your master security group, core/task security group, and additional security groups.

```
aws emr create-cluster --name "Test cluster" --release-label emr-4.2.0
--applications Name=Hue Name=Hive Name=Pig --use-default-roles --
ec2-attributes KeyName=myKey,ServiceAccessSecurityGroup=sg-service-
accessId,EmrManagedMasterSecurityGroup=sg-masterId,EmrManagedSlaveSecurityGroup=sg-
slaveId,AdditionalMasterSecurityGroups=securityGroupId,AdditionalSlaveSecurityGroups=securityGroupI
--instance-type m3.xlarge --instance-count 3
```

When you specify the instance count without using the `--instance-groups` parameter, a single master node is launched, and the remaining instances are launched as core nodes. All nodes use the instance type specified in the command.

2. To retrieve security group information for your cluster, type the following command and replace `j-1K48XXXXXXHCB` with your cluster ID:

```
aws emr describe-cluster --cluster-id j-1K48XXXXXXHCB
```

For more information, see [Amazon EMR commands in the AWS CLI](#).

Create SSH Credentials for the Master Node

Create an Amazon EC2 Key Pair and PEM File

Amazon EMR uses an Amazon EC2 key pair to ensure that you alone have access to the instances that you launch. The PEM file associated with this key pair is required to `ssh` directly to the master node of the cluster.

To create an Amazon EC2 key pair

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. Select a value for **Region**.
3. In the **Navigation** pane, choose **Key Pairs**.
4. On the **Key Pairs** page, choose **Create Key Pair**.
5. In the **Create Key Pair** dialog box, enter a name for your key pair, such as `mykeypair`.
6. Choose **Create**.
7. Save the resulting PEM file in a safe location.

Your Amazon EC2 key pair and an associated PEM file are created.

Modify Your PEM File

Amazon EMR enables you to work interactively with your cluster, allowing you to test cluster steps or troubleshoot your cluster environment. To log in directly to the master node of your running cluster, you can use `ssh` or PuTTY. Use your PEM file to authenticate to the master node. The PEM file requires a modification based on the tool you use that supports your operating system. Use the CLI to connect on Linux, UNIX, or Mac OS X computers or use PuTTY to connect on Microsoft Windows computers. For more information about how to install the Amazon EMR CLI or PuTTY, see the [Amazon EMR Getting Started Guide](#).

To modify your credentials file

- Create a local permissions file:
 - a. **Linux, UNIX, or Mac OS X**

Set the permissions on the PEM file or your Amazon EC2 key pair. For example, if you saved the file as `mykeypair.pem`, the command looks like the following:

```
chmod og-rwx mykeypair.pem
```

- b. Microsoft Windows
 - a. Download PuTTYgen.exe to your computer from <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>.
 - b. Launch PuTTYgen.
 - c. Choose **Load**, select the PEM file that you created earlier, and choose **Open**.
 - d. Choose **OK** on the **PuTTYgen Notice** telling you that the key was successfully imported.
 - e. Choose **Save private key** to save the key in the PPK format.
 - f. When PuTTYgen prompts you to save the key without a pass phrase, choose **Yes**.
 - g. Enter a name for your PuTTY private key, such as, `mykeypair.ppk`.
 - h. Choose **Save** to exit the PuTTYgen application.
 - i. Exit the PuTTYgen application.

Your credentials file is modified to allow you to log in directly to the master node of your running cluster.

Setting Permissions on the System Directory

You can set the permissions on the release directory by using a configuration to modify the `mapreduce.jobtracker.system.dir.permission` configuration variable. This is useful if you are running a custom configuration in which users other than "hadoop user" submit jobs to Hadoop.

To set permissions on the system directory

1. Create the following configuration, `myConfig.json` that sets the permissions for the directory using numerical permissions codes (octal notation). The following example configuration, with a permissions code of 777, gives full permissions to the user, group, and world. For more information, see [Octal notation](#) in Wikipedia.

```
[
  {
    "Classification": "mapred-site",
    "Properties": {
      "mapreduce.jobtracker.system.dir.permission": "777"
    }
  }
]
```

2. Submit this configuration using the AWS CLI:

```
aws emr create-cluster --release-label emr-4.0.0 --instance-type m3.xlarge --instance-count 2 --applications Name=Hadoop --configurations file:///./myConfig.json --use-default-roles
```

Configure Cluster Logging and Debugging

One of the things to decide as you plan your cluster is how much debugging support you want to make available. When you are first developing your data processing application, we recommend testing the

application on a cluster processing a small, but representative, subset of your data. When you do this, you will likely want to take advantage of all the debugging tools that Amazon EMR offers, such as archiving log files to Amazon S3.

When you've finished development and put your data processing application into full production, you may choose to scale back debugging. Doing so can save you the cost of storing log file archives in Amazon S3 and reduce processing load on the cluster as it no longer needs to write state to Amazon S3. The trade off, of course, is that if something goes wrong, you'll have fewer tools available to investigate the issue.

Default Log Files

By default, each cluster writes log files on the master node. These are written to the `/mnt/var/log/` directory. You can access them by using SSH to connect to the master node as described in [Connect to the Master Node Using SSH \(p. 173\)](#). Because these logs exist on the master node, when the node terminates—either because the cluster was shut down or because an error occurred—these log files are no longer available.

You do not need to enable anything to have log files written on the master node. This is the default behavior of Amazon EMR and Hadoop.

A cluster generates several types of log files, including:

- **Step logs** — These logs are generated by the Amazon EMR service and contain information about the cluster and the results of each step. The log files are stored in `/mnt/var/log/hadoop/steps/` directory on the master node. Each step logs its results in a separate numbered subdirectory: `/mnt/var/log/hadoop/steps/s-stepId1/` for the first step, `/mnt/var/log/hadoop/steps/s-stepId2/`, for the second step, and so on. The 13-character step identifiers (e.g. `stepId1`, `stepId2`) are unique to a cluster.
- **Hadoop and YARN component logs** — The logs for components associated with both Apache YARN and MapReduce, for example, are contained in separate folders in `/mnt/var/log/`. The log file locations for the Hadoop components under `/mnt/var/log` are as follows: `hadoop-hdfs`, `hadoop-mapreduce`, `hadoop-httpfs`, and `hadoop-yarn`. The `hadoop-state-pusher` directory is for the output of the Hadoop state pusher process.
- **Bootstrap action logs** — If your job uses bootstrap actions, the results of those actions are logged. The log files are stored in `/mnt/var/log/bootstrap-actions/` on the master node. Each bootstrap action logs its results in a separate numbered subdirectory: `/mnt/var/log/bootstrap-actions/1/` for the first bootstrap action, `/mnt/var/log/bootstrap-actions/2/`, for the second bootstrap action, and so on.
- **Instance state logs** — These logs provide information about the CPU, memory state, and garbage collector threads of the node. The log files are stored in `/mnt/var/log/instance-state/` on the master node.

Archive Log Files to Amazon S3

Note

You cannot currently use log aggregation to Amazon S3 with the `yarn logs` utility.

You can configure a cluster to periodically archive the log files stored on the master node to Amazon S3. This ensures that the log files are available after the cluster terminates, whether this is through normal shut down or due to an error. Amazon EMR archives the log files to Amazon S3 at 5 minute intervals.

To have the log files archived to Amazon S3, you must enable this feature when you launch the cluster. You can do this using the console, the CLI, or the API. By default, clusters launched using the console have log archiving enabled. For clusters launched using the CLI or API, logging to Amazon S3 must be manually enabled.

To archive log files to Amazon S3 using the console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**.
3. Choose **Go to advanced options**.
4. In the **Cluster Configuration** section, in the **Logging** field, accept the default option: **Enabled**.

This determines whether Amazon EMR captures detailed log data to Amazon S3. You can only set this when the cluster is created. For more information, see [View Log Files \(p. 146\)](#).

5. In the **Log folder S3 location** field, type (or browse to) an Amazon S3 path to store your logs. You may also allow the console to generate an Amazon S3 path for you. If you type the name of a folder that does not exist in the bucket, it is created.

When this value is set, Amazon EMR copies the log files from the EC2 instances in the cluster to Amazon S3. This prevents the log files from being lost when the cluster ends and the EC2 instances hosting the cluster are terminated. These logs are useful for troubleshooting purposes.

For more information, see [View Log Files \(p. 146\)](#).

6. Proceed with creating the cluster as described in [Plan and Configure Clusters \(p. 24\)](#).

To archive log files to Amazon S3 using the AWS CLI

To archive log files to Amazon S3 using the AWS CLI, type the `create-cluster` command and specify the Amazon S3 log path using the `--log-uri` parameter.

- To log files to Amazon S3 type the following command and replace `myKey` with the name of your EC2 key pair.

```
aws emr create-cluster --name "Test cluster" --release-label emr-4.0.0 --log-uri
s3://mybucket/logs/ --applications Name=Hadoop Name=Hive Name=Pig --use-default-roles
--ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3
```

When you specify the instance count without using the `--instance-groups` parameter, a single Master node is launched, and the remaining instances are launched as core nodes. All nodes will use the instance type specified in the command.

Note

If you have not previously created the default EMR service role and EC2 instance profile, type `aws emr create-default-roles` to create them before typing the `create-cluster` subcommand.

For more information on using Amazon EMR commands in the AWS CLI, see <http://docs.aws.amazon.com/cli/latest/reference/emr>.

To aggregate logs in Amazon S3 using the AWS CLI

Note

You cannot currently use log aggregation with the `yarn logs` utility. You can only use aggregation supported by this procedure.

Log aggregation (Hadoop 2.x) compiles logs from all containers for an individual application into a single file. To enable log aggregation to Amazon S3 using the AWS CLI, you use a bootstrap action at cluster launch to enable log aggregation and to specify the bucket to store the logs.

- **Important**

This setting has not worked in past 4.x releases of EMR. Please use releases greater than 4.3.0 if you want to configure this option.

To enable log aggregation create the following configuration file, `myConfig.json`, which contains the following:

```
[
  {
    "Classification": "yarn-site",
    "Properties": {
      "yarn.log-aggregation-enable": "true",
      "yarn.log-aggregation.retain-seconds": "-1",
      "yarn.nodemanager.remote-app-log-dir": "s3://\\mybucket\\logs"
    }
  }
]
```

Type the following command and replace `myKey` with the name of your EC2 key pair.

```
aws emr create-cluster --name "Test cluster" --release-label emr-4.5.0 --applications
Name=Hadoop --use-default-roles --ec2-attributes KeyName=myKey --instance-type
m3.xlarge --instance-count 3 --configurations file:///myConfig.json
```

When you specify the instance count without using the `--instance-groups` parameter, a single Master node is launched, and the remaining instances are launched as core nodes. All nodes will use the instance type specified in the command.

Note

If you have not previously created the default EMR service role and EC2 instance profile, type `aws emr create-default-roles` to create them before typing the `create-cluster` subcommand.

For more information on using Amazon EMR commands in the AWS CLI, see <http://docs.aws.amazon.com/cli/latest/reference/emr>.

Enable the Debugging Tool

The debugging tool is a graphical user interface that you can use to browse the log files from the console. When you enable debugging on a cluster, Amazon EMR archives the log files to Amazon S3 and then indexes those files. You can then use the graphical interface to browse the step, job, task, and task attempt logs for the cluster in an intuitive way.

To be able to use the graphical debugging tool, you must enable debugging when you launch the cluster. You can do this using the console, the CLI, or the API.

To enable the debugging tool using the Amazon EMR console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**.
3. Choose **Go to advanced options**.
4. In the **Cluster Configuration** section, in the **Logging** field, choose **Enabled**. You cannot enable debugging without enabling logging.
5. In the **Log folder S3 location** field, type an Amazon S3 path to store your logs.
6. In the **Debugging** field, choose **Enabled**.

The debugging option creates an Amazon SQS exchange to publish debugging messages to the Amazon EMR service backend. Charges for publishing messages to the exchange may apply. For more information, see <https://aws.amazon.com/sqs>.

7. Proceed with creating the cluster as described in [Plan and Configure Clusters \(p. 24\)](#).

To enable the debugging tool using the AWS CLI

To enable debugging using the AWS CLI, type the `create-cluster` subcommand with the `--enable-debugging` parameter. You must also specify the `--log-uri` parameter when enabling debugging.

- To enable debugging using the AWS CLI, type the following command and replace `myKey` with the name of your EC2 key pair.

```
aws emr create-cluster --name "Test cluster" --release-label emr-4.1.0 --log-uri
s3://mybucket/logs/ --enable-debugging --applications Name=Hadoop Name=Hive Name=Pig
--use-default-roles --ec2-attributes KeyName=myKey --instance-type m3.xlarge --
instance-count 3
```

When you specify the instance count without using the `--instance-groups` parameter, a single Master node is launched, and the remaining instances are launched as core nodes. All nodes will use the instance type specified in the command.

Note

If you have not previously created the default EMR service role and EC2 instance profile, type `aws emr create-default-roles` to create them before typing the `create-cluster` subcommand.

For more information on using Amazon EMR commands in the AWS CLI, see <http://docs.aws.amazon.com/cli/latest/reference/emr>.

Example Enabling debugging using the Java SDK

Enable debugging using the following StepConfig:

```
StepConfig stepConfig = new StepConfig()
    .withName("Enable Debugging")
    .withActionOnFailure("TERMINATE_JOB_FLOW")
    .withHadoopJarStep(new HadoopJarStepConfig()
    .withJar("command-runner.jar")
    .withArgs("state-pusher-script");
```

Debugging Option Information

Amazon EMR release 4.1 or later supports debugging in all regions.

The Amazon EMR creates an Amazon SQS queue to process debugging data. Message charges may apply. However, Amazon SQS does have Free Tier of up to 1,000,000 requests available. For more information, see the [Amazon SQS detail page](#).

Debugging requires the use of roles; your service role and instance profile must allow you to use all Amazon SQS API operations. If your roles are attached to Amazon EMR managed policies, you do not need to do anything to modify your roles. If you have custom roles, you need to add `sqs:*` permissions. For more information, see [Configure IAM Roles for Amazon EMR and Applications \(p. 106\)](#).

Tag Clusters

It can be convenient to categorize your AWS resources in different ways; for example, by purpose, owner, or environment. You can achieve this in Amazon EMR by assigning custom metadata to your Amazon

EMR clusters using tags. A tag consists of a key and a value, both of which you define. For Amazon EMR, the cluster is the resource-level that you can tag. For example, you could define a set of tags for your account's clusters that helps you track each cluster's owner or identify a production cluster versus a testing cluster. We recommend that you create a consistent set of tags to meet your organization requirements.

When you add a tag to an Amazon EMR cluster, the tag is also propagated to each active Amazon EC2 instance associated with the cluster. Similarly, when you remove a tag from an Amazon EMR cluster, that tag is removed from each associated active Amazon EC2 instance.

Important

Use the Amazon EMR console or CLI to manage tags on Amazon EC2 instances that are part of a cluster instead of the Amazon EC2 console or CLI, because changes that you make in Amazon EC2 do not synchronize back to the Amazon EMR tagging system.

You can identify an Amazon EC2 instance that is part of an Amazon EMR cluster by looking for the following system tags. In this example, *CORE* is the value for the instance group role and *j-12345678* is an example job flow (cluster) identifier value:

- `aws:elasticmapreduce:instance-group-role=CORE`
- `aws:elasticmapreduce:job-flow-id=j-12345678`

Note

Amazon EMR and Amazon EC2 interpret your tags as a string of characters with no semantic meaning.

You can work with tags using the AWS Management Console, the CLI, and the API.

You can add tags when creating a new Amazon EMR cluster and you can add, edit, or remove tags from a running Amazon EMR cluster. Editing a tag is a concept that applies to the Amazon EMR console, however using the CLI and API, to edit a tag you remove the old tag and add a new one. You can edit tag keys and values, and you can remove tags from a resource at any time a cluster is running. However, you cannot add, edit, or remove tags from a terminated cluster or terminated instances which were previously associated with a cluster that is still active. In addition, you can set a tag's value to the empty string, but you can't set a tag's value to null.

If you're using AWS Identity and Access Management (IAM) with your Amazon EC2 instances for resource-based permissions by tag, your IAM policies are applied to tags that Amazon EMR propagates to a cluster's Amazon EC2 instances. For Amazon EMR tags to propagate to your Amazon EC2 instances, your IAM policy for Amazon EC2 needs to allow permissions to call the Amazon EC2 `CreateTags` and `DeleteTags` APIs. Also, propagated tags can affect your Amazon EC2's resource-based permissions. Tags propagated to Amazon EC2 can be read as conditions in your IAM policy, just like other Amazon EC2 tags. Keep your IAM policy in mind when adding tags to your Amazon EMR clusters to avoid IAM users having incorrect permissions for a cluster. To avoid problems, make sure that your IAM policies do not include conditions on tags that you also plan to use on your Amazon EMR clusters. For more information, see [Controlling Access to Amazon EC2 Resources](#).

Tag Restrictions

The following basic restrictions apply to tags:

- The maximum number of tags per resource is 10.
- The maximum key length is 127 Unicode characters.
- The maximum value length is 255 Unicode characters.
- Tag keys and values are case sensitive.
- Do not use the `aws:` prefix in tag names and values because it is reserved for AWS use. In addition, you cannot edit or delete tag names or values with this prefix.

- You cannot change or edit tags on a terminated cluster.
- A tag value can be an empty string, but not null. In addition, a tag key cannot be an empty string.
- Keys and values can contain any alphabetic character in any language, any numeric character, white spaces, invisible separators, and the following symbols: `_ . : / = + - @`

For more information about tagging using the AWS Management Console, see [Working with Tags in the Console](#) in the *Amazon EC2 User Guide for Linux Instances*. For more information about tagging using the Amazon EC2API or command line, see [API and CLI Overview](#) in the *Amazon EC2 User Guide for Linux Instances*.

Tag Resources for Billing

You can use tags for organizing your AWS bill to reflect your own cost structure. To do this, sign up to get your AWS account bill with tag key values included. You can then organize your billing information by tag key values, to see the cost of your combined resources. Although Amazon EMR and Amazon EC2 have different billing statements, the tags on each cluster are also placed on each associated instance so you can use tags to link related Amazon EMR and Amazon EC2 costs.

For example, you can tag several resources with a specific application name, and then organize your billing information to see the total cost of that application across several services. For more information, see [Cost Allocation and Tagging](#) in the *AWS Billing and Cost Management User Guide*.

Add Tags to a New Cluster

You can add tags to a cluster while you are creating it.

To add tags when creating a new cluster using the console

1. In the Amazon EMR console, select the **Cluster List** page and click **Create cluster**.
2. On the **Create Cluster** page, in the **Tags** section, click the empty field in the **Key** column and type the name of your key.

When you begin typing the new tag, another tag line automatically appears to provide space for the next new tag.

3. Optionally, click the empty field in the **Value** column and type the name of your value.
4. Repeat the previous steps for each tag key/value pair to add to the cluster. When the cluster launches, any tags you enter are automatically associated with the cluster.

To add tags when creating a new cluster using the AWS CLI

The following example demonstrates how to add a tag to a new cluster using the AWS CLI. To add tags when you create a cluster, type the `create-cluster` subcommand with the `--tags` parameter.

- To add a tag named `costCenter` with key value `marketing` when you create a cluster, type the following command and replace `myKey` with the name of your EC2 key pair.

```
aws emr create-cluster --name "Test cluster" --release-label emr-4.0.0 --applications
  Name=Hadoop Name=Hive Name=Pig --tags "costCenter=marketing" --use-default-roles --
ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3
```

When you specify the instance count without using the `--instance-groups` parameter, a single Master node is launched, and the remaining instances are launched as core nodes. All nodes will use the instance type specified in the command.

Note

If you have not previously created the default EMR service role and EC2 instance profile, type `aws emr create-default-roles` to create them before typing the `create-cluster` subcommand.

For more information on using Amazon EMR commands in the AWS CLI, see <http://docs.aws.amazon.com/cli/latest/reference/emr>.

Adding Tags to an Existing Cluster

You can also add tags to an existing cluster.

To add tags to an existing cluster using the console

1. In the Amazon EMR console, select the **Cluster List** page and click a cluster to which to add tags.
2. On the **Cluster Details** page, in the **Tags** field, click **View All/Edit**.
3. On the **View All/Edit** page, click **Add**.
4. Click the empty field in the **Key** column and type the name of your key.
5. Optionally, click the empty field in the **Value** column and type the name of your value.
6. With each new tag you begin, another empty tag line appears under the tag you are currently editing. Repeat the previous steps on the new tag line for each tag to add.

To add tags to a running cluster using the AWS CLI

The following example demonstrates how to add tags to a running cluster using the AWS CLI. Type the `add-tags` subcommand with the `--tag` parameter to assign tags to a resource identifier (cluster ID). The resource ID is the cluster identifier available via the console or the `list-clusters` command.

Note

The `add-tags` subcommand currently accepts only one resource ID.

- To add two tags to a running cluster (one with a key named `production` with no value and the other with a key named `costCenter` with a value of `marketing`) type the following command and replace `j-KT4XXXXXXXXX1NM` with your cluster ID.

```
aws emr add-tags --resource-id j-KT4XXXXXXXXX1NM --tag "costCenter=marketing" --tag "other=accounting"
```

Note

When tags are added using the AWS CLI, there is no output from the command.

For more information on using Amazon EMR commands in the AWS CLI, see <http://docs.aws.amazon.com/cli/latest/reference/emr>.

View Tags on a Cluster

If you would like to see all the tags associated with a cluster, you can view them in the console or at the CLI.

To view the tags on a cluster using the console

1. In the Amazon EMR console, select the **Cluster List** page and click a cluster to view tags.

2. On the **Cluster Details** page, in the **Tags** field, some tags are displayed here. Click **View All/Edit** to display all available tags on the cluster.

To view the tags on a cluster using the AWS CLI

To view the tags on a cluster using the AWS CLI, type the `describe-cluster` subcommand with the `--query` parameter.

- To view a cluster's tags, type the following command and replace `j-KT4XXXXXXXX1NM` with your cluster ID.

```
aws emr describe-cluster --cluster-id j-KT4XXXXXXXX1NM --query Cluster.Tags
```

The output displays all the tag information about the cluster similar to the following:

Value: accounting	Value: marketing
Key: other	Key: costCenter

For more information on using Amazon EMR commands in the AWS CLI, see <http://docs.aws.amazon.com/cli/latest/reference/emr>.

Remove Tags from a Cluster

If you no longer need a tag, you can remove it from the cluster.

To remove tags from a cluster using the console

1. In the Amazon EMR console, select the **Cluster List** page and click a cluster from which to remove tags.
2. On the **Cluster Details** page, in the **Tags** field, click **View All/Edit**.
3. In the **View All/Edit** dialog box, click the **X** icon next to the tag to delete and click **Save**.
4. (Optional) Repeat the previous step for each tag key/value pair to remove from the cluster.

To remove tags from a cluster using the AWS CLI

To remove tags from a cluster using the AWS CLI, type the `remove-tags` subcommand with the `--tag-keys` parameter. When removing a tag, only the key name is required.

- To remove a tag from a cluster, type the following command and replace `j-KT4XXXXXXXX1NM` with your cluster ID.

```
aws emr remove-tags --resource-id j-KT4XXXXXXXX1NM --tag-keys "costCenter"
```

Note

You cannot currently remove multiple tags using a single command.

For more information on using Amazon EMR commands in the AWS CLI, see <http://docs.aws.amazon.com/cli/latest/reference/emr>.

Drivers and Third-Party Application Integration

You can run several popular big-data applications on Amazon EMR with utility pricing. This means you pay a nominal additional hourly fee for the third-party application while your cluster is running. It allows you to use the application without having to purchase an annual license. The following sections describe some of the tools you can use with EMR.

Topics

- [Use Business Intelligence Tools with Amazon EMR \(p. 129\)](#)
- [Using the MapR Distribution for Hadoop \(p. 129\)](#)

Use Business Intelligence Tools with Amazon EMR

You can use popular business intelligence tools like Microsoft Excel, MicroStrategy, QlikView, and Tableau with Amazon EMR to explore and visualize your data. Many of these tools require an ODBC (Open Database Connectivity) or JDBC (Java Database Connectivity) driver. You can download and install the necessary drivers from the links below:

- https://s3.amazonaws.com/amazon-odbc-jdbc-drivers/public/AmazonHiveJDBC_1.0.4.1004.zip
- https://s3.amazonaws.com/amazon-odbc-jdbc-drivers/public/AmazonHiveODBC_1.1.1.1001.zip
- <http://amazon-odbc-jdbc-drivers.s3.amazonaws.com/public/HBaseODBC.zip>

For more information about how you would connect a business intelligence tool like Microsoft Excel to Hive, go to http://cdn.simba.com/products/Hive/doc/Simba_Hive_ODBC_Quickstart.pdf.

Using the MapR Distribution for Hadoop

MapR is a third-party application offering an open, enterprise-grade distribution that makes Hadoop easier to use and more dependable. For ease of use, MapR provides network file system (NFS) and open database connectivity (ODBC) interfaces, a comprehensive management suite, and automatic compression. For dependability, MapR provides high availability with a self-healing no-NameNode architecture, and data protection with snapshots, disaster recovery, and with cross-cluster mirroring. For more information about MapR, go to <http://www.mapr.com/>.

There are several editions of MapR available on Amazon EMR:

- **M3 Edition** (versions 4.0.2, 3.1.1, 3.0.3, 3.0.2, 2.1.3)—The free version of a complete distribution for Hadoop. M3 delivers a fully random, read-write-capable platform that supports industry-standard interfaces (e.g., NFS, ODBC), and provides management, compression, and performance advantages.
- **M5 Edition** (versions 4.0.2, 3.1.1, 3.0.3, 3.0.2, 2.1.3)—The complete distribution for Apache Hadoop that delivers enterprise-grade features for all file operations on Hadoop. M5 features include mirroring, snapshots, NFS HA, and data placement control. For more information, including pricing, see [MapR on Amazon EMR Detail Page](#).
- **M7 Edition** (versions 4.0.2, 3.1.1, 3.0.3, 3.0.2)—The complete distribution for Apache Hadoop that delivers ease of use, dependability, and performance advantages for NoSQL and Hadoop applications. M7 provides scale, strong consistency, reliability, and continuous low latency with an architecture that does not require compactions or background consistency checks. For more information, including pricing, see [MapR on Amazon EMR Detail Page](#).

Note

For enterprise-grade reliability and consistent performance for Apache HBase applications, use the MapR M7 edition.

In addition, MapR does not support Ganglia and debugging and the MapR M3 and M5 editions on Amazon EMR do not support Apache HBase.
The version of MapR available in Amazon EMR that supports Hadoop 2.x is 4.0.2, and it is only available with Amazon EMR AMI 3.3.2

Launch an Amazon EMR cluster with MapR using the console

You can launch any standard cluster on a MapR distribution by specifying MapR when you set the Hadoop version.

To launch an Amazon EMR cluster with MapR using the console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**.
3. In the **Software Configuration** section, verify the fields according to the following table.

Important

If you launch a MapR job flow in a VPC, you must set `enableDnsHostnames` to true and all hosts in the cluster must have a fully qualified domain name. For more information, see [Using DNS with Your VPC](#) and [DHCP Options Sets](#) in the *Amazon VPC User Guide*.

Field	Action
Hadoop distribution	Choose MapR . This determines which version of Hadoop to run on your cluster. For more information about the MapR distribution for Hadoop, see Using the MapR Distribution for Hadoop (p. 129) .
Edition	Choose the MapR edition that meets your requirements. For more information, see Using the MapR Distribution for Hadoop (p. 129) .
Version	Choose the MapR version that meets your requirements. For more information, see Versions (p. 131) .
Arguments (Optional)	Specify any additional arguments to pass to MapR to meet your requirements. For more information, see Arguments (p. 132) .

4. Click **Done** and proceed with creating the cluster as described in [Plan and Configure Clusters \(p. 24\)](#).

Launch a Cluster with MapR Using the AWS CLI

To launch an Amazon EMR cluster with MapR using the AWS CLI

Note

You cannot yet launch MapR clusters when specifying the `--release-label` parameter.

To launch a cluster with MapR using the AWS CLI, type the `create-cluster` command with the `--applications` parameter.

- To launch a cluster with MapR, m7 edition, type the following command and replace *myKey* with the name of your EC2 key pair.
 - Linux, UNIX, and Mac OS X users:

```
aws emr create-cluster --name "MapR cluster" --applications Name=Hive Name=Pig \
Name=MapR,Args=--edition,m7,--version,4.0.2 --ami-version 3.3.2 \
--use-default-roles --ec2-attributes KeyName=myKey \
--instance-type m1.xlarge --instance-count 3
```

- Windows users:

```
aws emr create-cluster --name "MapR cluster" --applications Name=Hive Name=Pig
Name=MapR,Args=--edition,m7,--version,3.1.1 --ami-version 2.4 --use-default-roles --
ec2-attributes KeyName=myKey --instance-type m1.xlarge --instance-count 3
```

Note

The version of MapR available in Amazon EMR that supports Hadoop 2.x is 4.0.2, and it is only available with Amazon EMR AMI 3.3.2

When you specify the instance count without using the `--instance-groups` parameter, a single Master node is launched, and the remaining instances are launched as core nodes. All nodes will use the instance type specified in the command.

Note

If you have not previously created the default EMR service role and EC2 instance profile, type `aws emr create-default-roles` to create them before typing the `create-cluster` subcommand.

For more information on using Amazon EMR commands in the AWS CLI, see <http://docs.aws.amazon.com/cli/latest/reference/emr>.

MapR Editions, Versions, and Arguments

MapR supports several editions, versions, and arguments that you can pass to it using the Amazon EMR console or CLI. The following examples use the Amazon EMR CLI, however Amazon EMR provides an equivalent for each that you can use with the console.

For more information about launching clusters using the CLI, see the instructions for each cluster type in [Plan and Configure Clusters](#) (p. 24).

Editions

Using the CLI, you can pass the `--edition` parameter to specify the following editions:

- m3
- m5
- m7

The default edition is **m3** if not specified.

Versions

You can use `--version` to specify the following versions:

- M3 Edition - 4.0.2, 3.1.1, 3.0.3, 3.0.2, 2.1.3
- M5 Edition - 4.0.2, 3.1.1, 3.0.3, 3.0.2, 2.1.3
- M7 Edition - 4.0.2, 3.1.1, 3.0.3, 3.0.2

Note

MapR version 4.0.2 is available with AMI version 3.3.2 only.

Arguments

The `--supported-product` option supports optional arguments. Amazon EMR accepts and forwards the argument list to the corresponding installation script as bootstrap action arguments. The following example shows the syntax for using the `--supported-product` option with or without optional arguments. The fields `key1` and `value1`, etc. represent custom key/value pairs that MapR recognizes and the key named `edition` is shown among them as an example:

```
--supported-product mapr-m3
--supported-product mapr-m3 --args "--key1,value1,--key2,value2"
--supported-product mapr --args "--edition,m3,--key1,value1,--key2,value2"
--supported-product mapr-m3 --args "--edition,m3,--key1,value1,--key2,value2"
```

The following table lists the parameters that MapR reads when users specify them with the `--args` option.

Parameter	Description
<code>--edition</code>	The MapR edition.
<code>--version</code>	The MapR version.
<code>--owner-password</code>	The password for the Hadoop owner. The default is <code>hadoop</code> .
<code>--num-mapr-masters</code>	Any additional master nodes for m5/m7 clusters.
<code>--mfs-percentage</code>	The proportion of space from the attached storage volumes to be used for MapR file system; the default is 100 (all available space); the minimum is 50. Users who are doing large copies in and out of Amazon S3 storage using <code>s3distcp</code> or another such mechanism may see faster performance by allowing some of the storage space to be used as regular Linux storage. For example: <code>--mfs-percentage,90</code> . The remainder is mounted to <code>S3_TMP_DIR</code> as RAID0 file system.
<code>--hive-version</code>	The version of the Amazon Hive package. The default is <code>latest</code> . To disable the installation of Amazon Hive, use <code>--hive-version,none</code> .
<code>--pig-version</code>	The version of the Amazon Pig package. The default is <code>latest</code> . To disable the installation of Amazon Pig, use <code>--pig-version,none</code> .

The following table shows examples of commands, including arguments with the `--args` parameter, and which command MapR executes after interpreting the input.

Options	Commands Processed by MapR
<code>--supported-product mapr</code>	<code>--edition m3</code>

Options	Commands Processed by MapR
--supported-product mapr-m5	--edition m5
--supported-product mapr-m3	--edition m3
--with-supported-products mapr-m3 (deprecated)	--edition m3
--with-supported-products mapr-m5 (deprecated)	--edition m5
--supported-product mapr-m5 --args "--version,1.1"	--edition m5 --version 1.1
--supported-product mapr-m5 --args "--edition,m3"	N/A - returns an error
--supported-product mapr --args "--edition,m5"	--edition m5
--supported-product mapr --args "--version,1.1"	--edition m3 --version 1.1
--supported-product mapr --args "--edition,m5,--key1 value1"	--edition m5 --key1 value1

Note

The `--with-supported-products` option is deprecated and replaced by the `--supported-product` option, which provides the same Hadoop and MapR versions with added support for parameters.

Enabling MCS access for your Amazon EMR Cluster

After your MapR cluster is running, you need to open port 8453 to enable access to the MapR Control System (MCS) from hosts other than the host that launched the cluster. Follow these steps to open the port.

To open a port for MCS access

1. Select your job from the list of jobs displayed in **Your Elastic MapReduce Job Flows** in the **Elastic MapReduce** tab of the AWS Management Console, then select the **Description** tab in the lower pane. Make a note of the **Master Public DNS Name** value.
2. Click the **Amazon EC2** tab in the AWS Management Console to open the Amazon EC2 console.
3. In the navigation pane, select **Security Groups** under the **Network and Security** group.
4. In the **Security Groups** list, select **Elastic MapReduce-master**.
5. In the lower pane, click the **Inbound** tab.
6. In **Port Range**, type 8453. Leave the default value in the **Source** field.

Note

The standard MapR port is 8443. Use port number 8453 instead of 8443 when you use the MapR REST API calls to MapR on an Amazon EMR cluster.

7. Click **Add Rule**, then click **Apply Rule Changes**.
8. You can now navigate to the master node's DNS address. Connect to port 8453 to log in to the MapR Control System. Use the string `hadoop` for both login and password at the MCS login screen.

Note

For M5 MapR clusters on Amazon EMR, the MCS web server runs on the primary and secondary CLDB nodes, giving you another entry point to the MCS if the primary fails.

Testing Your Cluster

This section explains how to test your cluster by performing a word count on a sample input file.

To create a file and run a test job

1. Connect to the master node with SSH as user `hadoop`. Pass your `.pem` credentials file to `ssh` with the `-i` flag, as in this example:

```
ssh -i /path_to_pemfile/credentials.pem hadoop@masterDNS.amazonaws.com
```

2. Create a simple text file:

```
cd /mapr/MapR_EMR.amazonaws.com
mkdir in
echo "the quick brown fox jumps over the lazy dog" > in/data.txt
```

3. Run the following command to perform a word count on the text file:

```
hadoop jar /opt/mapr/hadoop/hadoop-0.20.2/hadoop-0.20.2-dev-examples.jar wordcount /
mapr/MapR_EMR.amazonaws.com/in/ /mapr/MapR_EMR.amazonaws.com/out/
```

As the job runs, you should see terminal output similar to the following:

```
12/06/09 00:00:37 INFO fs.JobTrackerWatcher: Current running JobTracker is:
ip10118194139.ec2.internal/10.118.194.139:9001
12/06/09 00:00:37 INFO input.FileInputFormat: Total input paths to process : 1
12/06/09 00:00:37 INFO mapred.JobClient: Running job: job_201206082332_0004
12/06/09 00:00:38 INFO mapred.JobClient: map 0% reduce 0%
12/06/09 00:00:50 INFO mapred.JobClient: map 100% reduce 0%
12/06/09 00:00:57 INFO mapred.JobClient: map 100% reduce 100%
12/06/09 00:00:58 INFO mapred.JobClient: Job complete: job_201206082332_0004
12/06/09 00:00:58 INFO mapred.JobClient: Counters: 25
12/06/09 00:00:58 INFO mapred.JobClient: Job Counters
12/06/09 00:00:58 INFO mapred.JobClient: Launched reduce tasks=1
12/06/09 00:00:58 INFO mapred.JobClient: Aggregate execution time of mappers(ms)=6193
12/06/09 00:00:58 INFO mapred.JobClient: Total time spent by all reduces waiting after
reserving slots (ms)=0
12/06/09 00:00:58 INFO mapred.JobClient: Total time spent by all maps waiting after
reserving slots (ms)=0
12/06/09 00:00:58 INFO mapred.JobClient: Launched map tasks=1
12/06/09 00:00:58 INFO mapred.JobClient: Datalocalmap tasks=1
12/06/09 00:00:58 INFO mapred.JobClient: Aggregate execution time of reducers(ms)=4875
12/06/09 00:00:58 INFO mapred.JobClient: FileSystemCounters
12/06/09 00:00:58 INFO mapred.JobClient: MAPRFS_BYTES_READ=385
12/06/09 00:00:58 INFO mapred.JobClient: MAPRFS_BYTES_WRITTEN=276
12/06/09 00:00:58 INFO mapred.JobClient: FILE_BYTES_WRITTEN=94449
12/06/09 00:00:58 INFO mapred.JobClient: MapReduce Framework
12/06/09 00:00:58 INFO mapred.JobClient: Map input records=1
12/06/09 00:00:58 INFO mapred.JobClient: Reduce shuffle bytes=94
12/06/09 00:00:58 INFO mapred.JobClient: Spilled Records=16
12/06/09 00:00:58 INFO mapred.JobClient: Map output bytes=80
12/06/09 00:00:58 INFO mapred.JobClient: CPU_MILLISECONDS=1530
12/06/09 00:00:58 INFO mapred.JobClient: Combine input records=9
12/06/09 00:00:58 INFO mapred.JobClient: SPLIT_RAW_BYTES=125
12/06/09 00:00:58 INFO mapred.JobClient: Reduce input records=8
12/06/09 00:00:58 INFO mapred.JobClient: Reduce input groups=8
12/06/09 00:00:58 INFO mapred.JobClient: Combine output records=8
12/06/09 00:00:58 INFO mapred.JobClient: PHYSICAL_MEMORY_BYTES=329244672
12/06/09 00:00:58 INFO mapred.JobClient: Reduce output records=8
12/06/09 00:00:58 INFO mapred.JobClient: VIRTUAL_MEMORY_BYTES=3252969472
```

```
12/06/09 00:00:58 INFO mapred.JobClient: Map output records=9
12/06/09 00:00:58 INFO mapred.JobClient: GC time elapsed (ms)=1
```

4. Check the `/mapr/MapR_EMR.amazonaws.com/out` directory for a file named `part-r-00000` with the results of the job.

```
cat out/part-r-00000
brown 1
dog 1
fox 1
jumps 1
lazy 1
over 1
quick 1
the 2
```

Tutorial: Launch an Amazon EMR Cluster with MapR M7

This tutorial guides you through launching an Amazon EMR cluster featuring the M7 edition of the [MapR distribution for Hadoop](#). The MapR distribution for Hadoop is a complete Hadoop distribution that provides many unique capabilities, such as industry-standard NFS and ODBC interfaces, end-to-end management, high reliability, and automatic compression. You can manage a MapR cluster through the web-based [MapR Control System](#), the command line, or a REST API. M7 provides enterprise-grade capabilities such as high availability, [snapshot](#) and [mirror volumes](#), and native MapR table functionality on MapR-FS, enabling responsive HBase-style flat table databases compatible with snapshots and mirroring. It provides a single platform for storing and processing unstructured and structured data, integrated with existing infrastructure, applications, and tools.

Note

To use the commands in this tutorial, download and install the AWS CLI. For more information see [Installing the AWS CLI](#) in the *AWS Command Line Interface User Guide*.

1. To launch a cluster with MapR, m7 edition, type the following command and replace `myKey` with the name of your EC2 key pair.
 - Linux, UNIX, and Mac OS X users:

```
aws emr create-cluster --name "MapR cluster" --applications Name=Hive Name=Pig \
Name=MapR,Args=--edition,m7,--version,4.0.2 --ami-version 3.3.2 \
--use-default-roles --ec2-attributes KeyName=myKey \
--instance-type m1.xlarge --instance-count 3
```

- Windows users:

```
aws emr create-cluster --name "MapR cluster" --applications Name=Hive Name=Pig
Name=MapR,Args=--edition,m7,--version,3.1.1 --ami-version 2.4 --use-default-roles --
ec2-attributes KeyName=myKey --instance-type m1.xlarge --instance-count 3
```

Note

The versions of MapR available in Amazon EMR do not currently support Hadoop 2.x. When specifying the `--ami-version`, use a Hadoop 1.x AMI.

When you specify the instance count without using the `--instance-groups` parameter, a single Master node is launched, and the remaining instances are launched as core nodes. All nodes will use the instance type specified in the command.

Note

If you have not previously created the default EMR service role and EC2 instance profile, type `aws emr create-default-roles` to create them before typing the `create-cluster` subcommand.

After you run the command, the cluster takes between five and ten minutes to start. The `aws emr list-clusters` command shows your cluster in the `STARTING` and `BOOTSTRAPPING` states before entering the `WAITING` state.

For more information on using Amazon EMR commands in the AWS CLI, see <http://docs.aws.amazon.com/cli/latest/reference/emr>.

2. Retrieve the cluster ID and then use SSH to connect to the cluster. In the following commands, replace `j-2AL4XXXXXX5T9` with your cluster ID, and replace `~/mykeypair.key` with the path to and filename of your key pair private key file.

```
aws emr list-clusters

aws emr ssh --cluster-id j-2AL4XXXXXX5T9 --key-pair-file ~/mykeypair.key
```

Note

For more information about accessing a cluster with SSH, see [Connect to the Master Node Using SSH \(p. 173\)](#).

3. MapR provides [volumes](#) as a way to organize data and manage cluster performance. A volume is a logical unit that allows you to apply policies to a set of files, directories, and sub-volumes. You can use volumes to enforce disk usage limits, set replication levels, establish ownership and accountability, and measure the cost generated by different projects or departments. Create a volume for each user, department, or project. You can mount volumes under other volumes to build a structure that reflects the needs of your organization. The volume structure defines how data is distributed across the nodes in your cluster.

Run the following command after connecting to your cluster over SSH to create a volume:

```
$ maprccli volume create -name tables -replicationtype low_latency -path /tables
```

4. The M7 Edition of the MapR distribution for Hadoop enables you to create and manipulate tables in many of the same ways that you create and manipulate files in a standard UNIX file system. In the M7 Edition, tables share the same namespace as files and are specified with full path names. You can create [mappings](#) between the table names used in Apache HBase and M7 Edition's native tables.

- a. Create a table with the following command:

```
$ echo "create '/tables/user_table', 'family' " | hbase shell
```

- b. List tables with the following command:

```
$ hadoop fs -ls /tables
Found 1 items
trwxr-xr-x 3 root root 2 2013-04-16 22:49 /tables/user_table

$ ls /mapr/MapR_EMR.amazonaws.com/tables
user_table
```

- c. Move or rename tables using the following command:

```
hadoop fs -mv /tables/user_table /tables/usertable
```

5. A snapshot is a read-only image of a volume at a specific point in time. Snapshots are useful any time you need to be able to roll back to a known good data set at a specific point in time.

- a. From the HBase shell, add a row to the newly-created table:

```
$ hbase shell
hbase(main):001:0> put '/tables/usertable', 'row_1' , 'family:child', 'username'
output: 0 row(s) in 0.0920 seconds
```

- b. Create the snapshot:

```
$ maprcli volume snapshot create -volume tables -snapshotname mysnap
```

- c. Change the table:

```
hbase(main):002:0> put '/tables/usertable', 'row_1' , 'family:location', 'San Jose'
```

- d. Snapshots are stored in a `.snapshot` directory. Scan the table from the snapshot and the current table to see the difference:

```
hbase shell >
scan '/tables/usertable'
ROW COLUMN+CELL
row_1 column=family:child, timestamp=1366154016042, value=username
row_1 column=family:home, timestamp=1366154055043, value=San Jose
1 row(s) in 0.2910 seconds
scan '/tables/.snapshot/mysnap/usertable'
ROW COLUMN+CELL
row_1 column=family:child, timestamp=1366154016042, value=username
1 row(s) in 0.2320 seconds
```

6. Test high availability:

- a. List the current regions on your system.

```
$ maprcli table region list -path /tables/usertable
secondarynodes scans primarynode puts
startkey gets lastheartbeat endkey
ip-10-191-5-21.ec2.internal, ip-10-68-37-140.ec2.internal ...
ip-10-4-74-175.ec2.internal ...
-INFINITY ... 0 INFINITY
```

- b. Restart the primary node for one of the regions. Make sure that the primary node is not the access point to the cluster. Restarting your access point will result in loss of cluster access and terminate your YCSB client.

Connect to the cluster with SSH and restart the node with the following command:

```
$ ssh -i /Users/username/downloads/MyKey_Context.pem
hadoop@ec2-23-20-100-174.compute-1.amazonaws.com
$ sudo /sbin/reboot
```

Note

The restart will take 15 to 30 seconds to complete.

- c. After the restart is complete, list your new regions to see that the former primary node is now listed as secondary.

```
$ maprcli table region list -path /tables/usertable
secondarynodes scans primarynode puts
```

```
startkey gets lastheartbeat endkey  
ip-10-191-5-21.ec2.internal, ip-10-68-37-140.ec2.internal ...  
ip-10-4-74-175.ec2.internal ...  
-INFINITY ... 0 INFINITY
```

7. To open the MapR Control System page, navigate to the address `https://hostname.compute-1.amazonaws.com:8453`. The username and password for the default installation are both **hadoop**. The URL for your node's hostname appears in the message-of-the-day that displays when you first log in to the node over SSH.

The Nodes view displays the nodes in the cluster, by rack. The Nodes view contains two panes: the Topology pane and the Nodes pane. The Topology pane shows the racks in the cluster. Selecting a rack displays that rack's nodes in the Nodes pane to the right. Selecting **Cluster** displays all the nodes in the cluster. Clicking any column name sorts data in ascending or descending order by that column. If your YCSB job is still running, you can see the put streams continuing from the Nodes view.

Manage Clusters

This documentation is for versions 4.x and 5.x of Amazon EMR. For information about Amazon EMR AMI versions 2.x and 3.x, see the [Amazon EMR Developer Guide \(PDF\)](#).

After you've launched your cluster, you can monitor and manage it. Amazon EMR provides several tools you can use to connect to and control your cluster.

Topics

- [View and Monitor a Cluster \(p. 139\)](#)
- [Connect to the Cluster \(p. 172\)](#)
- [Control Cluster Termination \(p. 185\)](#)
- [Scaling Cluster Resources \(p. 190\)](#)
- [Cloning a Cluster Using the Console \(p. 205\)](#)
- [Submit Work to a Cluster \(p. 206\)](#)
- [Automate Recurring Clusters with AWS Data Pipeline \(p. 210\)](#)

View and Monitor a Cluster

Amazon EMR provides several tools you can use to gather information about your cluster. You can access information about the cluster from the console, the CLI or programmatically. The standard Hadoop web interfaces and log files are available on the master node. You can also use monitoring services such as CloudWatch and Ganglia to track the performance of your cluster.

Topics

- [View Cluster Details \(p. 140\)](#)
- [Enhanced Step Debugging \(p. 144\)](#)
- [View Log Files \(p. 146\)](#)
- [View Cluster Instances in Amazon EC2 \(p. 150\)](#)
- [CloudWatch Events and Metrics \(p. 151\)](#)
- [View Cluster Application Metrics with Ganglia \(p. 171\)](#)
- [Logging Amazon EMR API Calls in AWS CloudTrail \(p. 171\)](#)

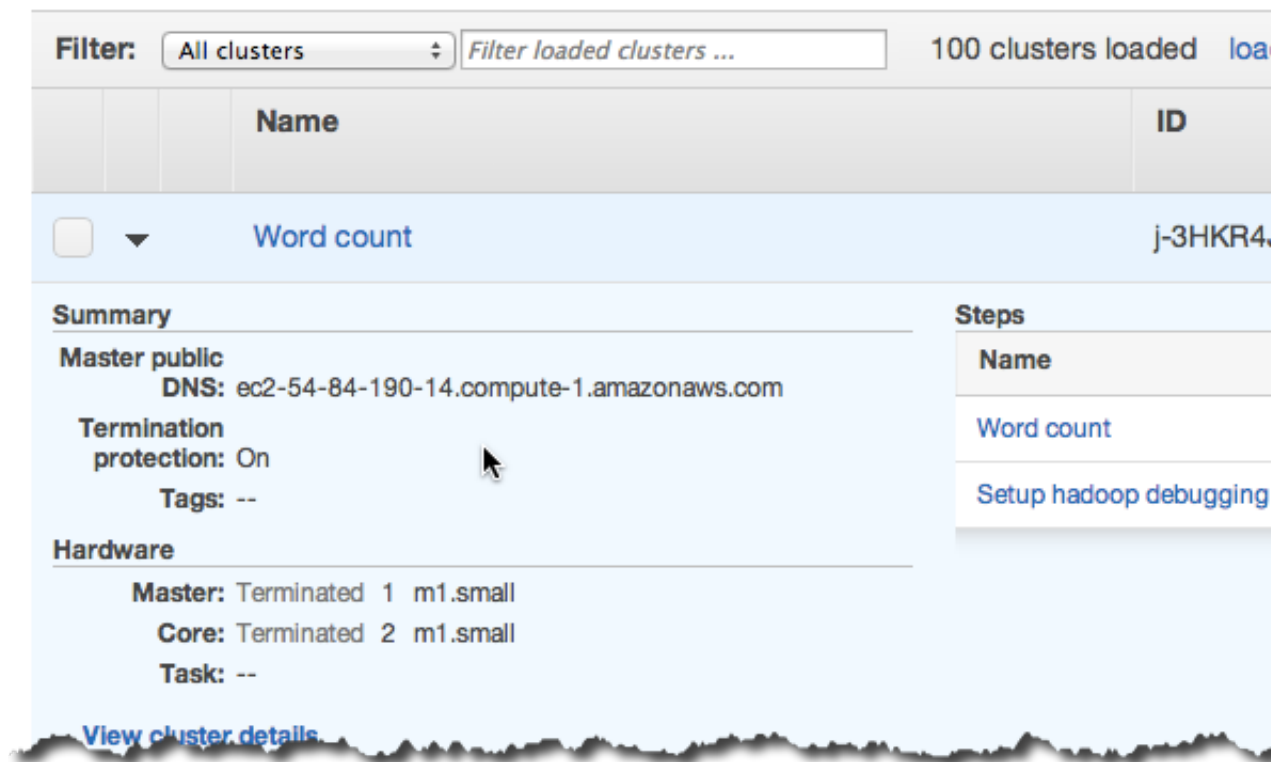
View Cluster Details

After you start a cluster, you can monitor its status and retrieve extended information about its execution. This section describes the methods used to view the details of Amazon EMR clusters. You can view clusters in any state.

This procedure explains how to view the details of a cluster using the Amazon EMR console.

To view cluster details using the console

1. Sign in to the AWS Management Console and open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. You have the option to view details in the **Cluster List** page. By selecting the arrow icon next to each cluster, the row view expands and gives some details and actions for the cluster you chose:



3. For more details, choose the cluster link to open the **Cluster Details** page. The **Summary** section displays detailed information about the selected cluster.

Summary: Word count

Status: Starting

Name: Word count

ID: j-3HJGUSHI3YN5P

Auto-terminate: No

Creation date: 2013-11-04 15:19:19 (local time - UTC-8)

End date: --

Master public DNS name: ec2-54-205-127-15.compute-1.amazonaws.com

Log URI: s3n://examples-bucket/

Applications:

Key name: --

Subnet ID: --

IAM role: --

Visible to all users: None

AMI version: 2.4.2

Hadoop distribution: Ama

Termination protection: On

Monitoring

Hardware Configuration

The following examples demonstrate how to retrieve cluster details using the AWS CLI and the Amazon EMR CLI.

To view cluster details using the AWS CLI

Use the `describe-cluster` command to view cluster-level details including status, hardware and software configuration, VPC settings, bootstrap actions, instance groups and so on.

- To use the `describe-cluster` command, you need the cluster ID. The cluster ID can be retrieved using the `list-clusters` command. To view cluster details, type the following command and replace `j-1K48XXXXXXHCB` with your cluster ID.

```
aws emr describe-cluster --cluster-id j-1K48XXXXXXHCB
```

The output is similar to the following:

```
{
  "Cluster": {
    "Status": {
      "Timeline": {
        "ReadyDateTime": 1438281058.061,
        "CreationDateTime": 1438280702.498
      },
      "State": "WAITING",
      "StateChangeReason": {
        "Message": "Waiting for steps to run"
      }
    },
    "Ec2InstanceAttributes": {
      "EmrManagedMasterSecurityGroup": "sg-cXXXXX0",
      "IamInstanceProfile": "EMR_EC2_DefaultRole",
      "Ec2KeyName": "myKey",
      "Ec2AvailabilityZone": "us-east-1c",

```

```
    "EmrManagedSlaveSecurityGroup": "sg-example"
  },
  "Name": "Development Cluster",
  "ServiceRole": "EMR_DefaultRole",
  "Tags": [],
  "TerminationProtected": false,
  "ReleaseLabel": "emr-4.0.0",
  "NormalizedInstanceHours": 16,
  "InstanceGroups": [
    {
      "RequestedInstanceCount": 1,
      "Status": {
        "Timeline": {
          "ReadyDateTime": 1438281058.101,
          "CreationDateTime": 1438280702.499
        },
        "State": "RUNNING",
        "StateChangeReason": {
          "Message": ""
        }
      },
      "Name": "CORE",
      "InstanceGroupType": "CORE",
      "Id": "ig-2EEXAMPLEEXP",
      "Configurations": [],
      "InstanceType": "m3.xlarge",
      "Market": "ON_DEMAND",
      "RunningInstanceCount": 1
    },
    {
      "RequestedInstanceCount": 1,
      "Status": {
        "Timeline": {
          "ReadyDateTime": 1438281023.879,
          "CreationDateTime": 1438280702.499
        },
        "State": "RUNNING",
        "StateChangeReason": {
          "Message": ""
        }
      },
      "Name": "MASTER",
      "InstanceGroupType": "MASTER",
      "Id": "ig-2A1234567XP",
      "Configurations": [],
      "InstanceType": "m3.xlarge",
      "Market": "ON_DEMAND",
      "RunningInstanceCount": 1
    }
  ],
  "Applications": [
    {
      "Version": "1.0.0",
      "Name": "Hive"
    },
    {
      "Version": "2.6.0",
      "Name": "Hadoop"
    },
    {
      "Version": "0.14.0",
      "Name": "Pig"
    },
    {
      "Version": "1.4.1",
      "Name": "Spark"
    }
  ]
}
```

```
    }  
  ],  
  "VisibleToAllUsers": true,  
  "BootstrapActions": [],  
  "MasterPublicDnsName": "ec2-X-X-X-X.compute-1.amazonaws.com",  
  "AutoTerminate": false,  
  "Id": "j-jobFlowID",  
  "Configurations": [  
    {  
      "Properties": {  
        "hadoop.security.groups.cache.secs": "250"  
      },  
      "Classification": "core-site"  
    },  
    {  
      "Properties": {  
        "mapreduce.tasktracker.reduce.tasks.maximum": "5",  
        "mapred.tasktracker.map.tasks.maximum": "2",  
        "mapreduce.map.sort.spill.percent": "90"  
      },  
      "Classification": "mapred-site"  
    },  
    {  
      "Properties": {  
        "hive.join.emit.interval": "1000",  
        "hive.merge.mapfiles": "true"  
      },  
      "Classification": "hive-site"  
    }  
  ]  
}
```

For more information on using Amazon EMR commands in the AWS CLI, see <http://docs.aws.amazon.com/cli/latest/reference/emr>.

To list clusters by creation date using the AWS CLI

To list clusters by creation date, type the `list-clusters` command with the `--created-after` and `--created-before` parameters.

- To list clusters created between 10-09-2014 and 10-12-2014, type the following command.

```
aws emr list-clusters --created-after 2014-10-09T00:12:00 --created-  
before 2014-10-12T00:12:00
```

The output lists all clusters created between the specified timestamps.

For more information on using Amazon EMR commands in the AWS CLI, see <http://docs.aws.amazon.com/cli/latest/reference/emr>.

Note

Amazon EMR preserves metadata information about completed clusters for your reference, at no charge, for two months. The console does not provide a way to delete completed clusters from the console; these are automatically removed for you after two months.

To list clusters by state using the AWS CLI

To list clusters by state, type the `list-clusters` command with the `--cluster-states` parameter. Valid cluster states include: `STARTING`, `BOOTSTRAPPING`, `RUNNING`, `WAITING`, `TERMINATING`, `TERMINATED`, and `TERMINATED_WITH_ERRORS`.

You can also use several shortcut parameters to view clusters. The `--active` parameter filters clusters in the `STARTING`, `BOOTSTRAPPING`, `RUNNING`, `WAITING`, or `TERMINATING` states. The `--terminated` parameter filters clusters in the `TERMINATED` state. The `--failed` parameter filters clusters in the `TERMINATED_WITH_ERRORS` state.

- Type the following command to list all `TERMINATED` clusters.

```
aws emr list-clusters --cluster-states TERMINATED
```

Or:

```
aws emr list-clusters --terminated
```

The output lists all clusters in the state specified.

For more information on using Amazon EMR commands in the AWS CLI, see <http://docs.aws.amazon.com/cli/latest/reference/emr>.

For more information about the input parameters unique to `DescribeJobFlows`, see [DescribeJobFlows](#).

Enhanced Step Debugging

If an Amazon EMR step fails and you submitted your work using the Step API operation with an AMI of version 5.x or later, Amazon EMR can identify and return the root cause of the step failure in some cases, along with the name of the relevant log file and a portion of the application stack trace via API. For example, the following failures can be identified:

- A common Hadoop error such as the output directory already exists, the input directory does not exist, or an application runs out of memory.
- Java errors such as an application that was compiled with an incompatible version of Java or run with a main class that is not found.
- An issue accessing objects stored in Amazon S3.

This information is available using the [DescribeStep](#) and [ListSteps](#) API operations. The [FailureDetails](#) field of the [StepSummary](#) returned by those operations. To access the FailureDetails information, use the AWS CLI, console, or AWS SDK.

To view failure details using the AWS Console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Cluster List** and select a cluster.
3. Select the arrow icon next to each step to view more details.

If the step has failed and Amazon EMR can identify the root cause, you see the details of the failure.

▼

[Add step](#) [Clone step](#)

Steps

Filter:

All steps



Filter steps ...

4 steps

ID	Name
s-IVXTXADABLBE	WordCount
<p>Status: FAILED</p> <p>Reason: Output directory already exists.</p> <p>Log File: s3://aws-logs-us-east-1/elasticmapreduce</p> <p>Details: org.apache.hadoop.mapred.FileAlreadyE</p> <p>JAR location: s3://bucket/hadoop-mapreduce-example</p> <p>Main class: None</p> <p>Arguments: wordcount s3://bucket/input/hello.txt s3:</p> <p>Action on failure: Continue</p>	
s-3R6M87MQTDGPS	WordCount
s-43HPMI46PES7	Custom JAR
s-21EQLTWVIUTUV	Setup hadoop debuggi

To view failure details using the AWS CLI

- To get failure details for a step using the AWS CLI, use the `describe-step` command.

```
aws emr describe-step --cluster-id j-1K48XXXXXHCB --step-id s-3QM0XXXXXXM1W
```

The output will look similar to the following:

```
{
  "Step": {
    "Status": {
      "FailureDetails": {
        "LogFile": "s3://myBucket/logs/j-1K48XXXXXHCb/steps/s-3QM0XXXXXM1W/stderr.gz",
        "Message": "org.apache.hadoop.mapred.FileAlreadyExistsException: Output
directory s3://myBucket/logs/beta already exists",
        "Reason": "Output directory already exists."
      },
      "Timeline": {
        "EndDateTime": 1469034209.143,
        "CreationDateTime": 1469033847.105,
        "StartDateTime": 1469034202.881
      },
      "State": "FAILED",
      "StateChangeReason": {}
    },
    "Config": {
      "Args": [
        "wordcount",
        "s3://myBucket/input/input.txt",
        "s3://myBucket/logs/beta"
      ],
      "Jar": "s3://myBucket/jars/hadoop-mapreduce-examples-2.7.2-amzn-1.jar",
      "Properties": {}
    },
    "Id": "s-3QM0XXXXXM1W",
    "ActionOnFailure": "CONTINUE",
    "Name": "ExampleJob"
  }
}
```

View Log Files

Amazon EMR and Hadoop both produce log files that report status on the cluster. By default, these are written to the master node in the `/mnt/var/log/` directory. Depending on how you configured your cluster when you launched it, these logs may also be archived to Amazon S3 and may be viewable through the graphical debugging tool.

There are many types of logs written to the master node. Amazon EMR writes step, bootstrap action, and instance state logs. Apache Hadoop writes logs to report the processing of jobs, tasks, and task attempts. Hadoop also records logs of its daemons. For more information about the logs written by Hadoop, go to <http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html>.

Topics

- [View Log Files on the Master Node \(p. 146\)](#)
- [View Log Files Archived to Amazon S3 \(p. 148\)](#)
- [View Log Files in the Debugging Tool \(p. 149\)](#)

View Log Files on the Master Node

The following table lists some of the log files you'll find on the master node.

Location	Description
/mnt/var/log/bootstrap-actions	Logs written during the processing of the bootstrap actions.
/mnt/var/log/hadoop-state-pusher	Logs written by the Hadoop state pusher process.
/mnt/var/log/instance-controller	Instance controller logs.
/mnt/var/log/instance-state	Instance state logs. These contain information about the CPU, memory state, and garbage collector threads of the node.
/mnt/var/log/service-nanny	Logs written by the service nanny process.
/mnt/var/log/ <i>application</i>	Logs specific to an application such as Hadoop, Spark, or Hive.
/mnt/var/log/hadoop/steps/ <i>N</i>	<p>Step logs that contain information about the processing of the step. The value of <i>N</i> indicates the stepId assigned by Amazon EMR. For example, a cluster has two steps: s-1234ABCDEFGH and s-5678IJKLMNOP. The first step is located in /mnt/var/log/hadoop/steps/s-1234ABCDEFGH/ and the second step in /mnt/var/log/hadoop/steps/s-5678IJKLMNOP/.</p> <p>The step logs written by Amazon EMR are as follows.</p> <ul style="list-style-type: none"> • controller — Information about the processing of the step. If your step fails while loading, you can find the stack trace in this log. • syslog — Describes the execution of Hadoop jobs in the step. • stderr — The standard error channel of Hadoop while it processes the step. • stdout — The standard output channel of Hadoop while it processes the step.

To view log files on the master node.

1. Use SSH to connect to the master node as described in [Connect to the Master Node Using SSH \(p. 173\)](#).
2. Navigate to the directory that contains the log file information you wish to view. The preceding table gives a list of the types of log files that are available and where you will find them. The following example shows the command for navigating to the step log with an ID, s-1234ABCDEFGH.

```
cd /mnt/var/log/hadoop/steps/s-1234ABCDEFGH/
```

3. Use a text editor installed on the master node to view the contents of the log file. There are several you can choose from: vi, nano, and emacs. The following example shows how to open the controller step log using the nano text editor.

```
nano controller
```


View Log Files Archived to Amazon S3

By default, Amazon EMR clusters launched using the console automatically archive log files to Amazon S3. You can specify your own log path, or you can allow the console to automatically generate a log path for you. For clusters launched using the CLI or API, you must configure Amazon S3 log archiving manually.

When Amazon EMR is configured to archive log files to Amazon S3, it stores the files in the S3 location you specified, in the `/JobFlowId/` folder, where `JobFlowId` is the cluster identifier.

The following table lists some of the log files you'll find on Amazon S3.

Location	Description
<code>/JobFlowId/node/</code>	Node logs, including bootstrap action, instance state, and application logs for the node. The logs for each node are stored in a folder labeled with the identifier of the EC2 instance of that node.
<code>/JobFlowId/node/instanceId/application</code>	The logs created by each application or daemon associated with an application. For example, the Hive server log is located at <code>JobFlowId/node/instanceId/hive/hive-server.log</code> .
<code>/JobFlowId/steps/N/</code>	<p>Step logs that contain information about the processing of the step. The value of <code>N</code> indicates the stepId assigned by Amazon EMR. For example, a cluster has two steps: <code>s-1234ABCDEFGH</code> and <code>s-5678IJKLMNOP</code>. The first step is located in <code>/mnt/var/log/hadoop/steps/s-1234ABCDEFGH/</code> and the second step in <code>/mnt/var/log/hadoop/steps/s-5678IJKLMNOP/</code>.</p> <p>The step logs written by Amazon EMR are as follows.</p> <ul style="list-style-type: none"> • controller — Information about the processing of the step. If your step fails while loading, you can find the stack trace in this log. • syslog — Describes the execution of Hadoop jobs in the step. • stderr — The standard error channel of Hadoop while it processes the step. • stdout — The standard output channel of Hadoop while it processes the step.
<code>/JobFlowId/containers</code>	Application container logs. The logs for each YARN application are stored in these locations.

To view log files archived to Amazon S3 using the console

1. Sign in to the AWS Management Console and open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. Open the S3 bucket specified when you configured the cluster to archive log files in Amazon S3.

3. Navigate to the log file containing the information to display. The preceding table gives a list of the types of log files that are available and where you will find them.
4. Double-click on a log file to view it in the browser.

If you don't want to view the log files in the Amazon S3 console, you can download the files from Amazon S3 to your local machine using a tool such as the Amazon S3 Organizer plug-in for the Firefox web browser, or by writing an application to retrieve the objects from Amazon S3. For more information, see [Getting Objects](#) in the *Amazon Simple Storage Service Developer Guide*.

View Log Files in the Debugging Tool

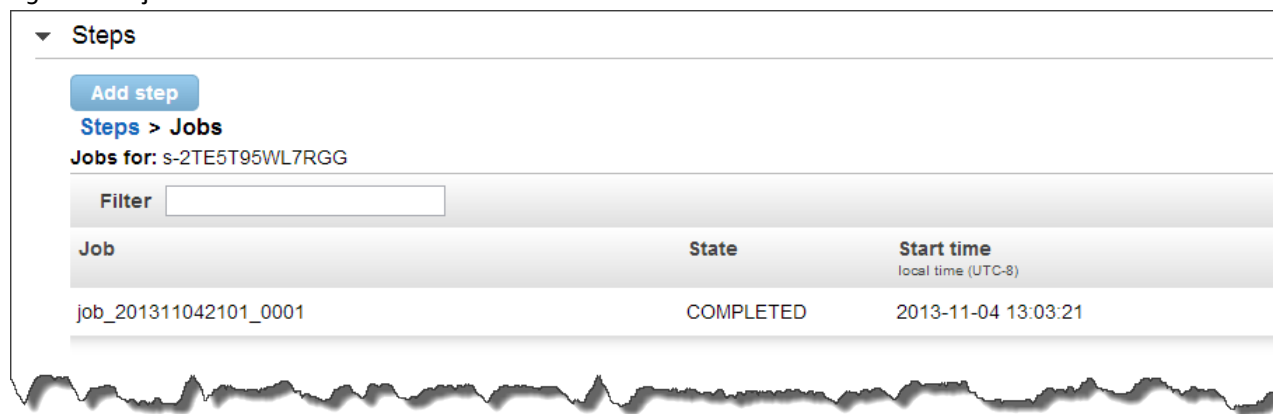
Amazon EMR does not automatically enable the debugging tool. You must configure this when you launch the cluster.

To view cluster logs using the console

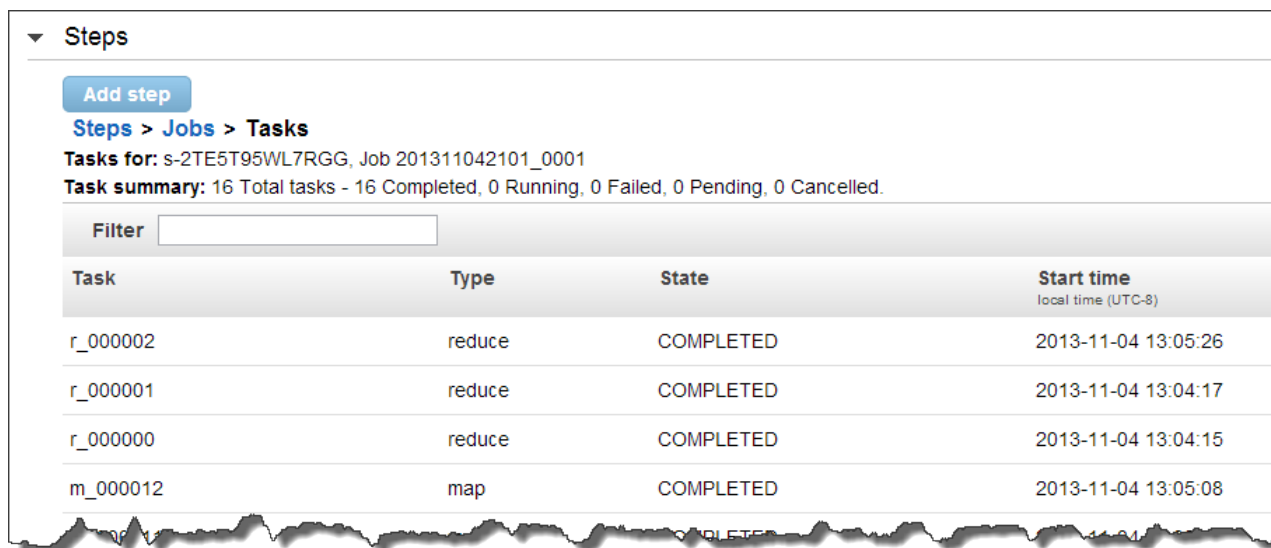
1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. From the **Cluster List** page, choose the details icon next to the cluster you want to view.

This brings up the **Cluster Details** page. In the **Steps** section, the links to the right of each step display the various types of logs available for the step. These logs are generated by Amazon EMR.

3. To view a list of the Hadoop jobs associated with a given step, choose the **View Jobs** link to the right of the step.
4. To view a list of the Hadoop tasks associated with a given job, choose the **View Tasks** link to the right of the job.



5. To view a list of the attempts a given task has run while trying to complete, choose the **View Attempts** link to the right of the task.



▼ Steps

[Add step](#)

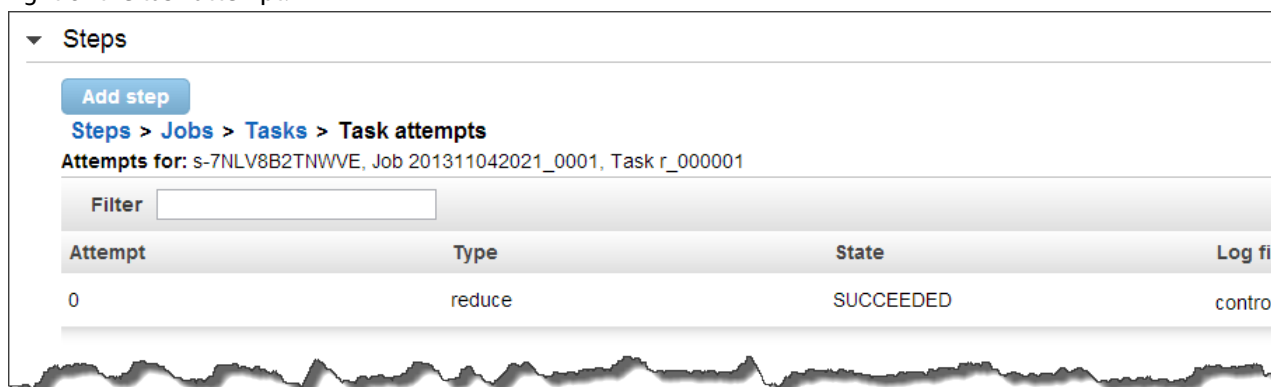
[Steps](#) > [Jobs](#) > [Tasks](#)

Tasks for: s-2TE5T95WL7RGG, Job 201311042101_0001

Task summary: 16 Total tasks - 16 Completed, 0 Running, 0 Failed, 0 Pending, 0 Cancelled.

Task	Type	State	Start time <small>(local time (UTC-8))</small>
r_000002	reduce	COMPLETED	2013-11-04 13:05:26
r_000001	reduce	COMPLETED	2013-11-04 13:04:17
r_000000	reduce	COMPLETED	2013-11-04 13:04:15
m_000012	map	COMPLETED	2013-11-04 13:05:08

6. To view the logs generated by a task attempt, choose the **stderr**, **stdout**, and **syslog** links to the right of the task attempt.



▼ Steps

[Add step](#)

[Steps](#) > [Jobs](#) > [Tasks](#) > [Task attempts](#)

Attempts for: s-7NLV8B2TNWVE, Job 201311042021_0001, Task r_000001

Attempt	Type	State	Log fi
0	reduce	SUCCEEDED	contro

The debugging tool displays links to the log files after Amazon EMR uploads the log files to your bucket on Amazon S3. Because log files are uploaded to Amazon S3 every 5 minutes, it can take a few minutes for the log file uploads to complete after the step completes.

Amazon EMR periodically updates the status of Hadoop jobs, tasks, and task attempts in the debugging tool. You can click **Refresh List** in the debugging panes to get the most up-to-date status of these items.

View Cluster Instances in Amazon EC2

To help you manage your resources, Amazon EC2 allows you to assign metadata to resources in the form of tags. Each Amazon EC2 tag consists of a key and a value. Tags allow you to categorize your Amazon EC2 resources in different ways: for example, by purpose, owner, or environment.

You can search and filter resources based on the tags. The tags assigned using your AWS account are available only to you. Other accounts sharing the resource cannot view your tags.

Amazon EMR automatically tags each EC2 instance it launches with key-value pairs that identify the cluster and the instance group to which the instance belongs. This makes it easy to filter your EC2 instances to show, for example, only those instances belonging to a particular cluster or to show all of

the currently running instances in the task-instance group. This is especially useful if you are running several clusters concurrently or managing large numbers of EC2 instances.

These are the predefined key-value pairs that Amazon EMR assigns:

Key	Value
aws:elasticmapreduce:job-flow-id	<job-flow-identifier>
aws:elasticmapreduce:instance-group-role	<group-role>

The values are further defined as follows:

- The <job-flow-identifier> is the ID of the cluster the instance is provisioned for. It appears in the format j-XXXXXXXXXXXXXX.
- The <group-role> is one of the following values: master, core, or task. These values correspond to the master instance group, core instance group, and task instance group.

You can view and filter on the tags that Amazon EMR adds. For more information, see [Using Tags](#) in the *Amazon EC2 User Guide for Linux Instances*. Because the tags set by Amazon EMR are system tags and cannot be edited or deleted, the sections on displaying and filtering tags are the most relevant.

Note

Amazon EMR adds tags to the EC2 instance when its status is updated to running. If there's a latency period between the time the EC2 instance is provisioned and the time its status is set to running, the tags set by Amazon EMR do not appear until the instance starts. If you don't see the tags, wait for a few minutes and refresh the view.

CloudWatch Events and Metrics

You can use events and metrics to track the activity and health of an Amazon EMR cluster, viewing events and metrics quickly in the Amazon EMR console for a single cluster, and viewing events for all clusters in a region. You can use CloudWatch Events to define an action to take when Amazon EMR generates an event that matches a pattern that you specify, and you can also use CloudWatch to monitor metrics.

Events are useful for monitoring a specific occurrence within a cluster—for example, when a cluster changes state from starting to running. Metrics are useful for monitoring a specific value—for example, the percentage of available disk space that HDFS is using within a cluster.

For more information about CloudWatch Events, see the [Amazon CloudWatch Events User Guide](#). For more information about CloudWatch metrics, see [Using Amazon CloudWatch Metrics](#) and [Creating Amazon CloudWatch Alarms](#) in the *Amazon CloudWatch User Guide*.

Topics

- [Monitor CloudWatch Events \(p. 151\)](#)
- [Monitor Metrics with CloudWatch \(p. 157\)](#)

Monitor CloudWatch Events

Amazon EMR tracks events and keeps information about them for up to seven days. Changes in the state of clusters, instance groups, automatic scaling policies, and steps cause an event to be recorded. Each event has information such as the date and time the event occurred, along with further detail about the event, such as the cluster or instance group affected.

The following table lists Amazon EMR events, along with the state or state change that the event indicates, the severity of the event, and event messages. Each event is represented as a JSON object that is sent automatically to an event stream. The JSON object includes further detail about the event. The JSON object is particularly important when you set up rules for event processing using CloudWatch Events because rules seek to match patterns in the JSON object. For more information, see [Events and Event Patterns](#) and [Amazon EMR Events](#) in the *Amazon CloudWatch Events User Guide*.

Cluster Events

State or State Change	Severity	Message
STARTING	INFO	Amazon EMR cluster <code>%s (%s)</code> was requested at <code>[time]</code> and is being created.
STARTING	INFO	<p>Note Applies only to clusters with the instance fleets configuration and multiple subnets selected within a VPC.</p> <p>Amazon EMR cluster <code>%s (%n)</code> is being created in subnet <code>(%s)</code> in VPC <code>(%)</code> in availability zone <code>(%)</code>, which was chosen from the specified VPC options.</p>
STARTING	INFO	<p>Note Applies only to clusters with the instance fleets configuration and multiple Availability Zones selected within EC2-Classic.</p> <p>Amazon EMR cluster <code>%s (%n)</code> is being created in availability zone <code>(%)</code>, which was chosen from the specified availability zone options.</p>
RUNNING	INFO	Amazon EMR cluster <code>%s (%s)</code> began running steps at <code>[time]</code> .
WAITING	INFO	<p>Amazon EMR cluster <code>%s (%s)</code> was created at <code>[time]</code> and is ready for use.</p> <p>—or—</p> <p>Amazon EMR cluster <code>%s (%s)</code> finished running all pending steps at <code>[time]</code>.</p>

State or State Change	Severity	Message
		Note A cluster in the <code>WAITING</code> state may nevertheless be processing jobs.
<code>TERMINATED</code>	The severity depends on the reason for the state change, as shown in the following: <ul style="list-style-type: none"> • CRITICAL if the cluster terminated with any of the following state change reasons: <code>INTERNAL_ERROR</code>, <code>VALIDATION_ERROR</code>, <code>INSTANCE_FAILURE</code>, <code>BOOTSTRAP_FAILURE</code>, or <code>STEP_FAILURE</code>. • INFO if the cluster terminated with any of the following state change reasons: <code>USER_REQUEST</code> or <code>ALL_STEPS_COMPLETED</code>. 	Amazon EMR Cluster <code>%s</code> (<code>%s</code>) has terminated at <code>[time]</code> with a reason of <code>[StateChangeReason:Code]</code> .
<code>TERMINATED_WITH_ERRORS</code>	CRITICAL	Amazon EMR Cluster <code>%s</code> (<code>%s</code>) has terminated with errors at <code>[time]</code> with a reason of <code>[StateChangeReason:Code]</code> .

Instance Fleet Events

Note

The instance fleets configuration is available only in Amazon EMR versions 4.8.0 and later, excluding 5.0.x versions.

State or State Change	Severity	Message
From <code>PROVISIONING</code> to <code>WAITING</code>	INFO	Provisioning for instance fleet <code>%s</code> in Amazon EMR cluster <code>%s</code> (<code>%n</code>) is complete. Provisioning started at <code>%t</code> and took <code>%d</code> minutes. The instance fleet now has On-Demand capacity of <code>num</code> and Spot capacity of <code>num</code> . Target On-Demand capacity was <code>num</code> , and target Spot capacity was <code>num</code> .
From <code>WAITING</code> to <code>RESIZING</code>	INFO	A resize for instance fleet <code>%s</code> in Amazon EMR cluster <code>%s</code> (<code>%n</code>) started at <code>%s</code> . The instance fleet is resizing from an On-Demand capacity of <code>num</code> to a target of <code>num</code> , and from a Spot capacity of <code>num</code> to a target of <code>num</code> .
From <code>RESIZING</code> to <code>WAITING</code>	INFO	The resizing operation for instance fleet <code>%s</code> in Amazon

State or State Change	Severity	Message
		EMR cluster %s (%n) is complete. The resize started at %t and took %d minutes. The instance fleet now has On-Demand capacity of XX and Spot capacity of YY. Target On-Demand capacity was XX and target Spot capacity was ZZ.
From RESIZING to WAITING	WARN	The resizing operation for instance fleet %s in Amazon EMR cluster %s (%n) has reached the timeout and stopped. The resize started at %t and stopped after %d minutes. The instance fleet now has On-Demand capacity of XX and Spot capacity of YY. Target On-Demand capacity was XX and target Spot capacity was ZZ.

Instance Group Events

State or State Change	Severity	Message
From RESIZING to RUNNING	INFO	The resizing operation for instance group %s in Amazon EMR cluster %s (%s) is complete. It now has an instance count of %d. The resize started at %s and took %d minutes to complete.
From RUNNING to RESIZING	INFO	A resize for instance group %s in Amazon EMR cluster %s (%s) started at %s. It is resizing from an instance count of %d to %d.

Automatic Scaling Policy Events

State or State Change	Severity	Message
PENDING	INFO	<p>An Auto Scaling policy was added to instance group %s in Amazon EMR cluster %s (%s) at [time]. The policy is pending attachment.</p> <p>—or—</p> <p>The Auto Scaling policy for instance group %s in Amazon EMR cluster %s (%s) was</p>

State or State Change	Severity	Message
		updated at <i>[time]</i> . The policy is pending attachment.
ATTACHED	INFO	The Auto Scaling policy for instance group <i>%s</i> in Amazon EMR cluster <i>%s (%s)</i> was attached at <i>[time]</i> .
DETACHED	INFO	The Auto Scaling policy for instance group <i>%s</i> in Amazon EMR cluster <i>%s (%s)</i> was detached at <i>[time]</i> .
FAILED	ERROR	<p>The Auto Scaling policy for instance group <i>%s</i> in Amazon EMR cluster <i>%s (%s)</i> could not attach and failed at <i>[time]</i>.</p> <p>—or—</p> <p>The Auto Scaling policy for instance group <i>%s</i> in Amazon EMR cluster <i>%s (%s)</i> could not detach and failed at <i>[time]</i>.</p>

Step Events

State or State Change	Severity	Message
PENDING	INFO	Step <i>[stepId] (step name)</i> was added to Amazon EMR cluster <i>%s (%s)</i> at <i>[time]</i> and is pending execution.
CANCEL_PENDING	WARN	Step <i>[stepId] (step name)</i> in Amazon EMR cluster <i>%s (%s)</i> was cancelled at <i>[time]</i> and is pending cancellation.
RUNNING	INFO	Step <i>[stepId] (step name)</i> in Amazon EMR cluster <i>%s (%s)</i> started running at <i>[time]</i> .
COMPLETED	INFO	Step <i>[stepId] (step name)</i> in Amazon EMR cluster <i>%s (%s)</i> completed execution at <i>[time]</i> . The step started running at <i>%s</i> and took <i>%d</i> minutes to complete.
CANCELLED	WARN	Cancellation request has succeeded for cluster step <i>[stepId] (step name)</i> in Amazon EMR cluster <i>%s (%s)</i>

State or State Change	Severity	Message
		at <i>[time]</i> , and the step is now cancelled.
FAILED	ERROR	Step <i>[stepId]</i> (<i>step name</i>) in Amazon EMR cluster <i>%s (%s)</i> failed at <i>[time]</i> .

Viewing Events Using the Amazon EMR Console

For each cluster, you can view a simple list of events in the details pane, which lists events in descending order of occurrence. You can also view all events for all clusters in a region in descending order of occurrence.

Note

If you don't want a user to see all cluster events for a region, add a statement that denies permission ("Effect": "Deny") for the `elasticmapreduce:ViewEventsFromAllClustersInConsole` action to a policy that is attached to the user.

To view events for all clusters in a region

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Events**.

To view events for a particular cluster

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Cluster List**, select a cluster, and then choose **View details**.
3. Choose **Events** in the cluster details pane.

The screenshot shows the Amazon EMR console interface. The left sidebar contains navigation links: Amazon EMR, Cluster list, Security configurations, VPC subnets, and Help. The main content area is titled 'Cluster: JeffGoliAutoScale' and shows various configuration details like Connections, Tags, Summary, Configuration Details, Network and Hardware, and Security and Access. The 'Events' tab is selected, showing a table of events.

Time	Event Description	Resource ID	Resource Type	Event Type	Severity	Full Date & Time
Dec 14 12:15 PM	The resize for your Amazon EMR cluster j-30YFAD070EDM (JeffGoliAutoScale) is complete and instance group ig-14DCCMR2T12 is now in state 'Running'. The resize was initiated at 2016-12-14 20:34 UTC and took 101 minutes to complete.	ig-14DCCMR2T12	Instance Group	Instance Group State Change	INFO	December 14, 2016 at 02:15:55 PM (UTC-8)
Dec 14 12:35 PM	A resize for your Amazon EMR cluster j-30YFAD070EDM (JeffGoliAutoScale) was started at 2016-12-14 20:34 UTC. Instance group ig-14DCCMR2T12 RESIZING from instance count of 2 to 1.	ig-14DCCMR2T12	Instance Group	Instance Group State Change	INFO	December 14, 2016 at 12:35:17 PM (UTC-8)

Creating Rules for Amazon EMR Events Using CloudWatch

Amazon EMR automatically sends events to a CloudWatch event stream. You can create rules that match events according to a specified pattern, and route the events to targets to take action, such as sending

an email notification. Patterns are matched against the event JSON object. For more information about Amazon EMR event details, see [Amazon EMR Events](#) in the *Amazon CloudWatch Events User Guide*.

To create a rule for an Amazon EMR event using the CloudWatch console

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Rules**, **Create rule**.
3. For **Event source**, choose **Amazon EMR**.
4. Choose event states and other details according to your requirements for event handling. To create a rule by modifying the JSON according to the guidelines in [Events and Event Patterns](#), choose **Show advanced options**, **edit**.
5. Select a target and add additional targets according to your requirements for event handling.
6. Choose **Configure details**, provide rule definition details, and then choose **Create rule**.

Monitor Metrics with CloudWatch

Metrics are updated every five minutes and automatically collected and pushed to CloudWatch for every EMR cluster. This interval is not configurable. There is no charge for the Amazon EMR metrics reported in CloudWatch. Metrics are archived for two weeks, after which the data is discarded.

How Do I Use Amazon EMR Metrics?

The metrics reported by Amazon EMR provide information that you can analyze in different ways. The table below shows some common uses for the metrics. These are suggestions to get you started, not a comprehensive list. For the complete list of metrics reported by Amazon EMR, see [Metrics Reported by Amazon EMR in CloudWatch \(p. 161\)](#).

How do I?	Relevant Metrics
Track the progress of my cluster	Look at the <code>RunningMapTasks</code> , <code>RemainingMapTasks</code> , <code>RunningReduceTasks</code> , and <code>RemainingReduceTasks</code> metrics.
Detect clusters that are idle	The <code>IsIdle</code> metric tracks whether a cluster is live, but not currently running tasks. You can set an alarm to fire when the cluster has been idle for a given period of time, such as thirty minutes.
Detect when a node runs out of storage	The <code>HDFSUtilization</code> metric is the percentage of disk space currently used. If this rises above an acceptable level for your application, such as 80% of capacity used, you may need to resize your cluster and add more core nodes.

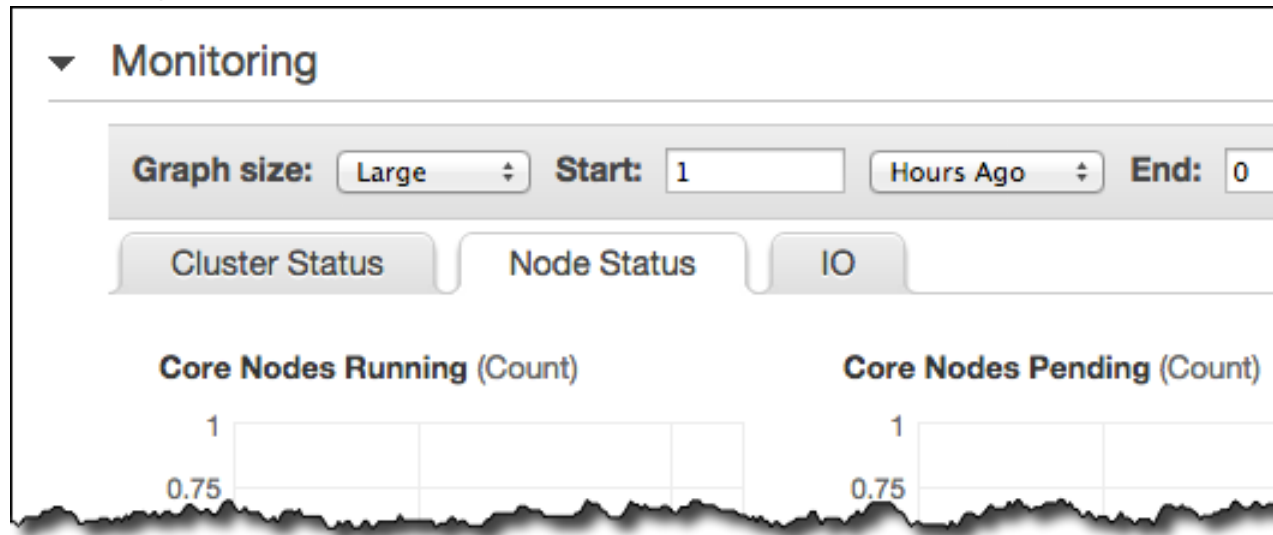
Accessing CloudWatch Metrics

There are many ways to access the metrics that Amazon EMR pushes to CloudWatch. You can view them through either the Amazon EMR console or CloudWatch console, or you can retrieve them using the CloudWatch CLI or the CloudWatch API. The following procedures show you how to access the metrics using these various tools.

To view metrics in the Amazon EMR console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.

2. To view metrics for a cluster, select a cluster to display the **Summary** pane.
3. Choose **Monitoring** to view information about that cluster. Choose any one of the tabs named **Cluster Status**, **Map/Reduce**, **Node Status**, **IO**, or **HBase** to load the reports about the progress and health of the cluster.
4. After you choose a metric to view, you can select a graph size. Edit **Start** and **End** fields to filter the metrics to a specific time frame.



To view metrics in the CloudWatch console

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **EMR**.
3. Scroll down to the metric to graph. You can search on the cluster identifier of the cluster to monitor.

Dashboard

Alarms

ALARM

INSUFFICIENT

OK

Billing

Metrics

Selected Metrics

DynamoDB

EBS

EC2

EMR

RDS

Redshift

Browse Metrics

Search Metrics

EMR Metrics

Showing the first 200 matching metrics. 4 additional metrics not listed for *EMR Metrics*.
Browse Metrics button above.

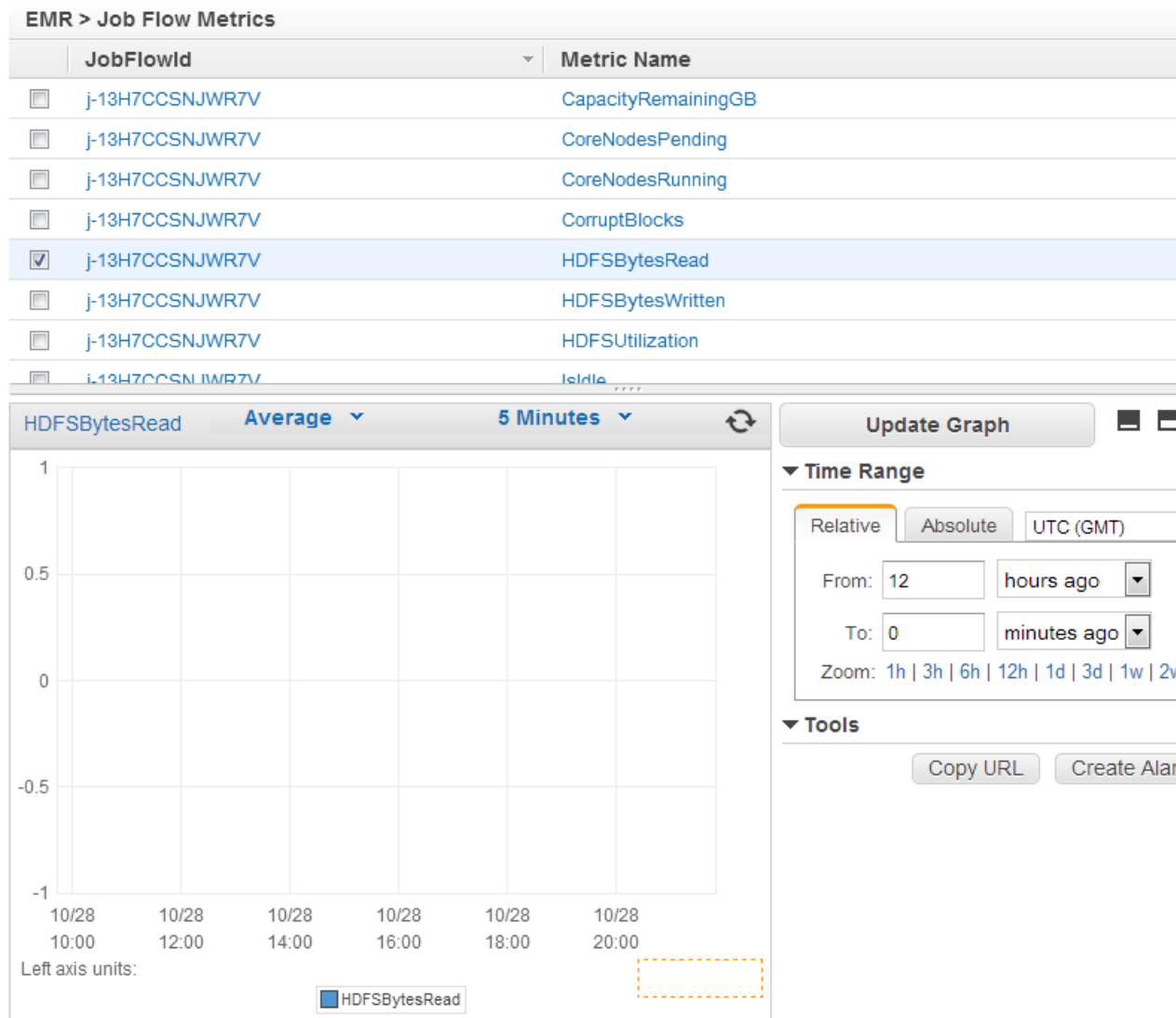
Select All | Clear

EMR > Job Flow Metrics

	JobFlowId	Metric Name
<input type="checkbox"/>	j-13H7CCSNJWR7V	CapacityRemainingGB
<input type="checkbox"/>	j-13H7CCSNJWR7V	CoreNodesPending
<input type="checkbox"/>	j-13H7CCSNJWR7V	CoreNodesRunning
<input type="checkbox"/>	j-13H7CCSNJWR7V	CorruptBlocks
<input type="checkbox"/>	j-13H7CCSNJWR7V	HDFSBytesRead
<input type="checkbox"/>	j-13H7CCSNJWR7V	HDFSBytesWritten
<input type="checkbox"/>	j-13H7CCSNJWR7V	HDFSUtilization
<input type="checkbox"/>	j-13H7CCSNJWR7V	IsIdle
<input type="checkbox"/>	j-13H7CCSNJWR7V	JobsFailed
<input type="checkbox"/>	j-13H7CCSNJWR7V	JobsRunning
<input type="checkbox"/>	j-13H7CCSNJWR7V	LiveDataNodes
<input type="checkbox"/>	j-13H7CCSNJWR7V	LiveTaskTrackers
<input type="checkbox"/>	j-13H7CCSNJWR7V	MapSlotsOpen
<input type="checkbox"/>	j-13H7CCSNJWR7V	MissingBlocks

- Open a metric to display the graph.

Select All | Clear



To access metrics from the CloudWatch CLI

- Call `mon-get-stats`. For more information, see the [Amazon CloudWatch User Guide](#).

To access metrics from the CloudWatch API

- Call `GetMetricStatistics`. For more information, see [Amazon CloudWatch API Reference](#).

Setting Alarms on Metrics

Amazon EMR pushes metrics to CloudWatch, which means you can use CloudWatch to set alarms on your Amazon EMR metrics. You can, for example, configure an alarm in CloudWatch to send you an email any time the HDFS utilization rises above 80%.

The following topics give you a high-level overview of how to set alarms using CloudWatch. For detailed instructions, see [Create or Edit a CloudWatch Alarm](#) in the *Amazon CloudWatch User Guide*.

Set alarms using the CloudWatch console

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. Choose **Create Alarm**. This launches the **Create Alarm Wizard**.
3. Choose **EMR Metrics** and scroll through the Amazon EMR metrics to locate the metric you want to place an alarm on. An easy way to display just the Amazon EMR metrics in this dialog box is to search on the cluster identifier of your cluster. Select the metric to create an alarm on and choose **Next**.
4. Fill in the **Name**, **Description**, **Threshold**, and **Time** values for the metric.
5. If you want CloudWatch to send you an email when the alarm state is reached, in the **Whenever this alarm:** field, choose **State is ALARM**. For **Send notification to:**, select an existing SNS topic. If you choose **Create topic**, you can set the name and email addresses for a new email subscription list. This list is saved and appears in the field for future alarms.

Note

If you use **Create topic** to create a new Amazon SNS topic, the email addresses must be verified before they receive notifications. Emails are only sent when the alarm enters an alarm state. If this alarm state change happens before the email addresses are verified, they do not receive a notification.

6. At this point, the **Define Alarm** screen gives you a chance to review the alarm that you're about to create. Choose **Create Alarm**.

Note

For more information about how to set alarms using the CloudWatch console, see [Create an Alarm that Sends Email](#) in the *Amazon CloudWatch User Guide*.

To set an alarm using the CloudWatch API

- Call `mon-put-metric-alarm`. For more information, see [Amazon CloudWatch User Guide](#).

To set an alarm using the CloudWatch API

- Call `PutMetricAlarm`. For more information, see [Amazon CloudWatch API Reference](#)

Metrics Reported by Amazon EMR in CloudWatch

The following table lists all the metrics that Amazon EMR reports in the console and pushes to CloudWatch.

Amazon EMR Metrics

Amazon EMR sends data for several metrics to CloudWatch. All Amazon EMR clusters automatically send metrics in five-minute intervals. Metrics are archived for two weeks; after that period, the data is discarded.

The `AWS/ElasticMapReduce` namespace includes the following metrics.

Note

Amazon EMR pulls metrics from a cluster. If a cluster becomes unreachable, no metrics are reported until the cluster becomes available again.

The following are Hadoop 1 metrics:

Metric	Description
<i>Cluster Status</i>	
IsIdle	<p>Indicates that a cluster is no longer performing work, but is still alive and accruing charges. It is set to 1 if no tasks are running and no jobs are running, and set to 0 otherwise. This value is checked at five-minute intervals and a value of 1 indicates only that the cluster was idle when checked, not that it was idle for the entire five minutes. To avoid false positives, you should raise an alarm when this value has been 1 for more than one consecutive 5-minute check. For example, you might raise an alarm on this value if it has been 1 for thirty minutes or longer.</p> <p>Use case: Monitor cluster performance</p> <p>Units: <i>Boolean</i></p>
JobsRunning	<p>The number of jobs in the cluster that are currently running.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
JobsFailed	<p>The number of jobs in the cluster that have failed.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
<i>Map/Reduce</i>	
MapTasksRunning	<p>The number of running map tasks for each job. If you have a scheduler installed and multiple jobs running, multiple graphs are generated.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
MapTasksRemaining	<p>The number of remaining map tasks for each job. If you have a scheduler installed and multiple jobs running, multiple graphs are generated. A remaining map task is one that is not in any of the following states: Running, Killed, or Completed.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
MapSlotsOpen	<p>The unused map task capacity. This is calculated as the maximum number of map tasks for a given cluster, less the total number of map tasks currently running in that cluster.</p> <p>Use case: Analyze cluster performance</p> <p>Units: <i>Count</i></p>
RemainingMapTasksPerSlot	<p>The ratio of the total map tasks remaining to the total map slots available in the cluster.</p>

Metric	Description
	<p>Use case: Analyze cluster performance</p> <p>Units: <i>Ratio</i></p>
ReduceTasksRunning	<p>The number of running reduce tasks for each job. If you have a scheduler installed and multiple jobs running, multiple graphs are generated.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
ReduceTasksRemaining	<p>The number of remaining reduce tasks for each job. If you have a scheduler installed and multiple jobs running, multiple graphs are generated.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
ReduceSlotsOpen	<p>Unused reduce task capacity. This is calculated as the maximum reduce task capacity for a given cluster, less the number of reduce tasks currently running in that cluster.</p> <p>Use case: Analyze cluster performance</p> <p>Units: <i>Count</i></p>
<i>Node Status</i>	
CoreNodesRunning	<p>The number of core nodes working. Data points for this metric are reported only when a corresponding instance group exists.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
CoreNodesPending	<p>The number of core nodes waiting to be assigned. All of the core nodes requested may not be immediately available; this metric reports the pending requests. Data points for this metric are reported only when a corresponding instance group exists.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
LiveDataNodes	<p>The percentage of data nodes that are receiving work from Hadoop.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Percent</i></p>
TaskNodesRunning	<p>The number of task nodes working. Data points for this metric are reported only when a corresponding instance group exists.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>

Metric	Description
TaskNodesPending	<p>The number of core nodes waiting to be assigned. All of the task nodes requested may not be immediately available; this metric reports the pending requests. Data points for this metric are reported only when a corresponding instance group exists.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
LiveTaskTrackers	<p>The percentage of task trackers that are functional.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Percent</i></p>
<i>IO</i>	
S3BytesWritten	<p>The number of bytes written to Amazon S3.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: <i>Bytes</i></p>
S3BytesRead	<p>The number of bytes read from Amazon S3.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: <i>Bytes</i></p>
HDFSUtilization	<p>The percentage of HDFS storage currently used.</p> <p>Use case: Analyze cluster performance</p> <p>Units: <i>Percent</i></p>
HDFSBytesRead	<p>The number of bytes read from HDFS.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: <i>Bytes</i></p>
HDFSBytesWritten	<p>The number of bytes written to HDFS.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: <i>Bytes</i></p>
MissingBlocks	<p>The number of blocks in which HDFS has no replicas. These might be corrupt blocks.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>

Metric	Description
TotalLoad	<p>The total number of concurrent data transfers.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
<i>HBase</i>	
BackupFailed	<p>Whether the last backup failed. This is set to 0 by default and updated to 1 if the previous backup attempt failed. This metric is only reported for HBase clusters.</p> <p>Use case: Monitor HBase backups</p> <p>Units: <i>Count</i></p>
MostRecentBackupDuration	<p>The amount of time it took the previous backup to complete. This metric is set regardless of whether the last completed backup succeeded or failed. While the backup is ongoing, this metric returns the number of minutes after the backup started. This metric is only reported for HBase clusters.</p> <p>Use case: Monitor HBase Backups</p> <p>Units: <i>Minutes</i></p>
TimeSinceLastSuccessfulBackup	<p>The number of elapsed minutes after the last successful HBase backup started on your cluster. This metric is only reported for HBase clusters.</p> <p>Use case: Monitor HBase backups</p> <p>Units: <i>Minutes</i></p>

The following metrics are available for Hadoop 2 AMIs:

Metric	Description
<i>Cluster Status</i>	
IsIdle	<p>Indicates that a cluster is no longer performing work, but is still alive and accruing charges. It is set to 1 if no tasks are running and no jobs are running, and set to 0 otherwise. This value is checked at five-minute intervals and a value of 1 indicates only that the cluster was idle when checked, not that it was idle for the entire five minutes. To avoid false positives, you should raise an alarm when this value has been 1 for more than one consecutive 5-minute check. For example, you might raise an alarm on this value if it has been 1 for thirty minutes or longer.</p> <p>Use case: Monitor cluster performance</p> <p>Units: <i>Boolean</i></p>
ContainerAllocated	<p>The number of resource containers allocated by the ResourceManager.</p>

Metric	Description
	<p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
ContainerReserved	<p>The number of containers reserved.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
ContainerPending	<p>The number of containers in the queue that have not yet been allocated.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
ContainerPendingRatio	<p>The ratio of pending containers to containers allocated ($\text{ContainerPendingRatio} = \text{ContainerPending} / \text{ContainerAllocated}$). If $\text{ContainerAllocated} = 0$, then $\text{ContainerPendingRatio} = \text{ContainerPending}$. The value of <code>ContainerPendingRatio</code> represents a number, not a percentage. This value is useful for scaling cluster resources based on container allocation behavior.</p>
AppsCompleted	<p>The number of applications submitted to YARN that have completed.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
AppsFailed	<p>The number of applications submitted to YARN that have failed to complete.</p> <p>Use case: Monitor cluster progress, Monitor cluster health</p> <p>Units: <i>Count</i></p>
AppsKilled	<p>The number of applications submitted to YARN that have been killed.</p> <p>Use case: Monitor cluster progress, Monitor cluster health</p> <p>Units: <i>Count</i></p>
AppsPending	<p>The number of applications submitted to YARN that are in a pending state.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
AppsRunning	<p>The number of applications submitted to YARN that are running.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>

Metric	Description
AppsSubmitted	<p>The number of applications submitted to YARN.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
<i>Node Status</i>	
CoreNodesRunning	<p>The number of core nodes working. Data points for this metric are reported only when a corresponding instance group exists.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
CoreNodesPending	<p>The number of core nodes waiting to be assigned. All of the core nodes requested may not be immediately available; this metric reports the pending requests. Data points for this metric are reported only when a corresponding instance group exists.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
LiveDataNodes	<p>The percentage of data nodes that are receiving work from Hadoop.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Percent</i></p>
MRTotalNodes	<p>The number of nodes presently available to MapReduce jobs.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
MRActiveNodes	<p>The number of nodes presently running MapReduce tasks or jobs.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
MRLostNodes	<p>The number of nodes allocated to MapReduce that have been marked in a LOST state.</p> <p>Use case: Monitor cluster health, Monitor cluster progress</p> <p>Units: <i>Count</i></p>
MRUnhealthyNodes	<p>The number of nodes available to MapReduce jobs marked in an UNHEALTHY state.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>

Metric	Description
MRDecommissionedNodes	<p>The number of nodes allocated to MapReduce applications that have been marked in a DECOMMISSIONED state.</p> <p>Use case: Monitor cluster health, Monitor cluster progress</p> <p>Units: <i>Count</i></p>
MRR rebootedNodes	<p>The number of nodes available to MapReduce that have been rebooted and marked in a REBOOTED state.</p> <p>Use case: Monitor cluster health, Monitor cluster progress</p> <p>Units: <i>Count</i></p>
<i>IO</i>	
S3BytesWritten	<p>The number of bytes written to Amazon S3.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: <i>Bytes</i></p>
S3BytesRead	<p>The number of bytes read from Amazon S3.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: <i>Bytes</i></p>
HDFSUtilization	<p>The percentage of HDFS storage currently used.</p> <p>Use case: Analyze cluster performance</p> <p>Units: <i>Percent</i></p>
HDFSBytesRead	<p>The number of bytes read from HDFS.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: <i>Bytes</i></p>
HDFSBytesWritten	<p>The number of bytes written to HDFS.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: <i>Bytes</i></p>
MissingBlocks	<p>The number of blocks in which HDFS has no replicas. These might be corrupt blocks.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>

Metric	Description
CorruptBlocks	<p>The number of blocks that HDFS reports as corrupted.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
TotalLoad	<p>The total number of concurrent data transfers.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
MemoryTotalMB	<p>The total amount of memory in the cluster.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Bytes</i></p>
MemoryReservedMB	<p>The amount of memory reserved.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Bytes</i></p>
MemoryAvailableMB	<p>The amount of memory available to be allocated.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Bytes</i></p>
YARNMemoryAvailablePercentage	<p>The percentage of remaining memory available to YARN ($\text{YARNMemoryAvailablePercentage} = \text{MemoryAvailableMB} / \text{MemoryTotalMB}$). This value is useful for scaling cluster resources based on YARN memory usage.</p>
MemoryAllocatedMB	<p>The amount of memory allocated to the cluster.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Bytes</i></p>
PendingDeletionBlocks	<p>The number of blocks marked for deletion.</p> <p>Use case: Monitor cluster progress, Monitor cluster health</p> <p>Units: <i>Count</i></p>
UnderReplicatedBlocks	<p>The number of blocks that need to be replicated one or more times.</p> <p>Use case: Monitor cluster progress, Monitor cluster health</p> <p>Units: <i>Count</i></p>

Metric	Description
DfsPendingReplicationBlocks	<p>The status of block replication: blocks being replicated, age of replication requests, and unsuccessful replication requests.</p> <p>Use case: Monitor cluster progress, Monitor cluster health</p> <p>Units: <i>Count</i></p>
CapacityRemainingGB	<p>The amount of remaining HDFS disk capacity.</p> <p>Use case: Monitor cluster progress, Monitor cluster health</p> <p>Units: <i>Bytes</i></p>
<i>HBase</i>	
HbaseBackupFailed	<p>Whether the last backup failed. This is set to 0 by default and updated to 1 if the previous backup attempt failed. This metric is only reported for HBase clusters.</p> <p>Use case: Monitor HBase backups</p> <p>Units: <i>Count</i></p>
MostRecentBackupDuration	<p>The amount of time it took the previous backup to complete. This metric is set regardless of whether the last completed backup succeeded or failed. While the backup is ongoing, this metric returns the number of minutes after the backup started. This metric is only reported for HBase clusters.</p> <p>Use case: Monitor HBase Backups</p> <p>Units: <i>Minutes</i></p>
TimeSinceLastSuccessfulBackup	<p>The number of elapsed minutes after the last successful HBase backup started on your cluster. This metric is only reported for HBase clusters.</p> <p>Use case: Monitor HBase backups</p> <p>Units: <i>Minutes</i></p>

Dimensions for Amazon EMR Metrics

Amazon EMR data can be filtered using any of the dimensions in the following table.

Dimension	Description
JobFlowId	The same as cluster ID, which is the unique identifier of a cluster in the form j-xxxxxxxxxxxxx. Find this value by clicking on the cluster in the Amazon EMR console.
JobId	The identifier of a job within a cluster. You can use this to filter the metrics returned from a cluster down to those that apply to a single job within the cluster. JobId takes the form job_XXXXXXXXXXXXX_XXXX.

View Cluster Application Metrics with Ganglia

Ganglia is available with Amazon EMR releases 4.2 and above. Ganglia is an open source project which is a scalable, distributed system designed to monitor clusters and grids while minimizing the impact on their performance. When you enable Ganglia on your cluster, you can generate reports and view the performance of the cluster as a whole, as well as inspect the performance of individual node instances. Ganglia is also configured to ingest and visualize Hadoop and Spark metrics. For more information, see the [Amazon EMR Release Guide](#).

Logging Amazon EMR API Calls in AWS CloudTrail

Amazon EMR is integrated with AWS CloudTrail, a service that captures API calls made by or on behalf of your AWS account. This information is collected and written to log files that are stored in an Amazon S3 bucket that you specify. API calls are logged when you use the Amazon EMR API, the Amazon EMR console, a back-end console, or the AWS CLI. Using the information collected by CloudTrail, you can determine what request was made to Amazon EMR, the source IP address the request was made from, who made the request, when it was made, and so on.

To learn more about CloudTrail, including how to configure and enable it, see the [AWS CloudTrail User Guide](#)

Topics

- [Amazon EMR Information in CloudTrail](#) (p. 171)
- [Understanding Amazon EMR Log File Entries](#) (p. 171)

Amazon EMR Information in CloudTrail

If CloudTrail logging is turned on, calls made to all Amazon EMR actions are captured in log files. All of the Amazon EMR actions are documented in the [Amazon EMR API Reference](#). For example, calls to the **ListClusters**, **DescribeCluster**, and **RunJobFlow** actions generate entries in CloudTrail log files.

Every log entry contains information about who generated the request. For example, if a request is made to create and run a new job flow (**RunJobFlow**), CloudTrail logs the user identity of the person or service that made the request. The user identity information helps you determine whether the request was made with root or IAM user credentials, with temporary security credentials for a role or federated user, or by another AWS service. For more information about CloudTrail fields, see [CloudTrail Event Reference](#) in the *AWS CloudTrail User Guide*.

You can store your log files in your bucket for as long as you want, but you can also define Amazon S3 lifecycle rules to archive or delete log files automatically. By default, your log files are encrypted by using Amazon S3 server-side encryption (SSE).

Understanding Amazon EMR Log File Entries

CloudTrail log files can contain one or more log entries composed of multiple JSON-formatted events. A log entry represents a single request from any source and includes information about the requested action, any input parameters, the date and time of the action, and so on. The log entries do not appear in any particular order. That is, they do not represent an ordered stack trace of the public API calls.

The following log file record shows that an IAM user called the **RunJobFlow** action by using the SDK.

```
{
```



```

"Records": [
{
    "eventVersion": "1.01",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/temporary-user-xx-7M",
        "accountId": "123456789012",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "temporary-user-xx-7M"
    },
    "eventTime": "2014-03-31T17:59:21Z",
    "eventSource": "elasticmapreduce.amazonaws.com",
    "eventName": "RunJobFlow",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "aws-sdk-java/unknown-version Linux/xx Java_HotSpot(TM)_64-
Bit_Server_VM/xx",
    "requestParameters": {
        "tags": [
            {
                "value": "prod",
                "key": "domain"
            },
            {
                "value": "us-east-1",
                "key": "realm"
            },
            {
                "value": "VERIFICATION",
                "key": "executionType"
            }
        ],
        "instances": {
            "slaveInstanceType": "m1.large",
            "ec2KeyName": "emr-integtest",
            "instanceCount": 1,
            "masterInstanceType": "m1.large",
            "keepJobFlowAliveWhenNoSteps": true,
            "terminationProtected": false
        },
        "visibleToAllUsers": false,
        "name": "Integ 1xm1large",
        "amiVersion": "3.0.4"
    },
    "responseElements": {
        "jobFlowId": "j-2WDJCGEG4E6AJ"
    },
    "requestID": "2f482daf-b8fe-11e3-89e7-75a3d0e071c5",
    "eventID": "b348a38d-f744-4097-8b2a-e68c9b424698"
},
...additional entries
]
}

```

Connect to the Cluster

When you run an Amazon EMR cluster, often all you need to do is run an application to analyze your data and then collect the output from an Amazon S3 bucket. At other times, you may want to interact with the master node while the cluster is running. For example, you may want to connect to the master node to run interactive queries, check log files, debug a problem with the cluster, monitor performance using

an application such as Ganglia that runs on the master node, and so on. The following sections describe techniques that you can use to connect to the master node.

In an EMR cluster, the master node is an Amazon EC2 instance that coordinates the EC2 instances that are running as task and core nodes. The master node exposes a public DNS name that you can use to connect to it. By default, Amazon EMR creates security group rules for master and slave nodes that determine how you access the nodes. For example, the master node security group contains a rule that allows you to connect to the master node using an SSH client over TCP port 22.

Note

You can connect to the master node only while the cluster is running. When the cluster terminates, the EC2 instance acting as the master node is terminated and is no longer available. To connect to the master node, you must also specify an Amazon EC2 key pair private key when you launch the cluster. The key pair private key provides the credentials for the SSH connection to the master node. If you launch a cluster from the console, the Amazon EC2 key pair private key is specified in the **Security and Access** section on the **Create Cluster** page.

By default, the ElasticMapReduce-master security group does not permit inbound SSH access. You may need to add an inbound rule that allows SSH access (TCP port 22) from the sources you want to have access. For more information about modifying security group rules, see [Adding Rules to a Security Group](#) in the *Amazon EC2 User Guide for Linux Instances*.

Important

Do not modify the remaining rules in the ElasticMapReduce-master security group. Modifying these rules may interfere with the operation of the cluster.

Topics

- [Connect to the Master Node Using SSH \(p. 173\)](#)
- [View Web Interfaces Hosted on Amazon EMR Clusters \(p. 177\)](#)

Connect to the Master Node Using SSH

Secure Shell (SSH) is a network protocol you can use to create a secure connection to a remote computer. After you make a connection, the terminal on your local computer behaves as if it is running on the remote computer. Commands you issue locally run on the remote computer, and the command output from the remote computer appears in your terminal window.

When you use SSH with AWS, you are connecting to an EC2 instance, which is a virtual server running in the cloud. When working with Amazon EMR, the most common use of SSH is to connect to the EC2 instance that is acting as the master node of the cluster.

Using SSH to connect to the master node gives you the ability to monitor and interact with the cluster. You can issue Linux commands on the master node, run applications such as Hive and Pig interactively, browse directories, read log files, and so on. You can also create a tunnel in your SSH connection to view the web interfaces hosted on the master node. For more information, see [View Web Interfaces Hosted on Amazon EMR Clusters \(p. 177\)](#).

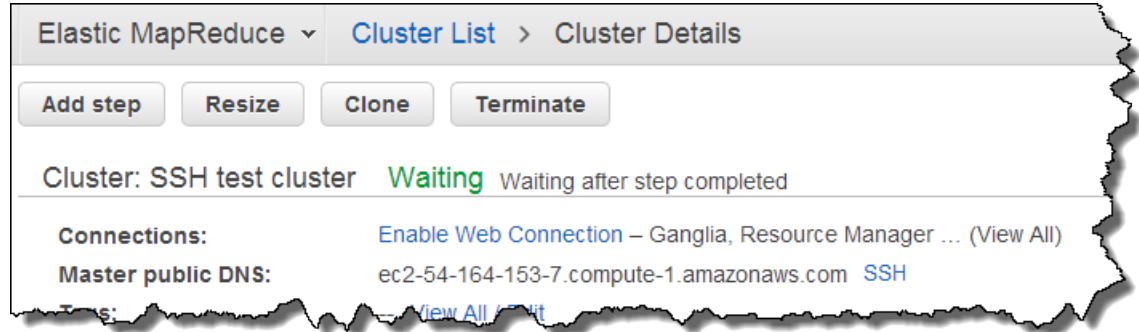
To connect to the master node using SSH, you need the public DNS name of the master node and your Amazon EC2 key pair private key. The Amazon EC2 key pair private key is specified when you launch the cluster. If you launch a cluster from the console, the Amazon EC2 key pair private key is specified in the **Security and Access** section on the **Create Cluster** page. For more information about accessing your key pair, see [Amazon EC2 Key Pairs](#) in the *Amazon EC2 User Guide for Linux Instances*.

Retrieve the Public DNS Name of the Master Node

You can retrieve the master public DNS name using the Amazon EMR console and the AWS CLI.

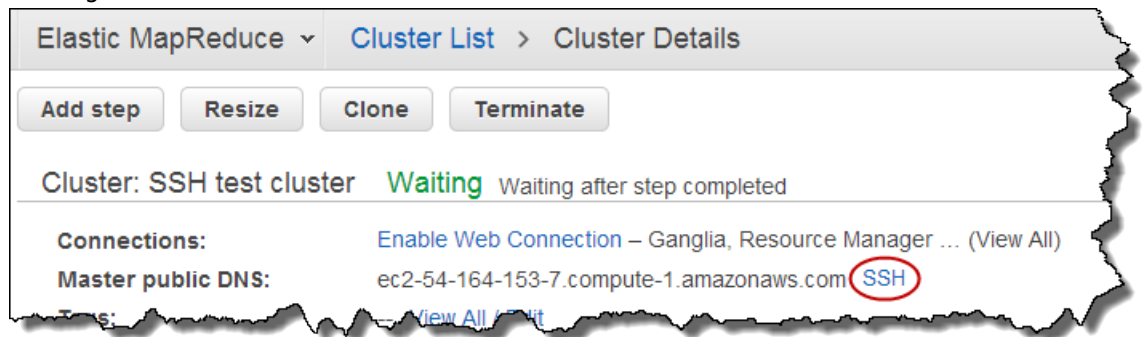
To retrieve the public DNS name of the master node using the Amazon EMR console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. On the **Cluster List** page, select the link for your cluster.
3. Note the **Master public DNS** value that appears at the top of the **Cluster Details** page.



Note

You may also choose the **SSH** link beside the master public DNS name for instructions on creating an SSH connection with the master node.



To retrieve the public DNS name of the master node using the AWS CLI

1. To retrieve the cluster identifier, type the following command.

```
aws emr list-clusters
```

The output lists your clusters including the cluster IDs. Note the cluster ID for the cluster to which you are connecting.

```
"Status": {
  "Timeline": {
    "ReadyDateTime": 1408040782.374,
    "CreationDateTime": 1408040501.213
  },
  "State": "WAITING",
  "StateChangeReason": {
    "Message": "Waiting after step completed"
  }
},
"NormalizedInstanceHours": 4,
"Id": "j-2AL4XXXXXX5T9",
```

```
"Name": "My cluster"
```

2. To list the cluster instances including the master public DNS name for the cluster, type one of the following commands. Replace `j-2AL4XXXXXX5T9` with the cluster ID returned by the previous command.

```
aws emr list-instances --cluster-id j-2AL4XXXXXX5T9
```

Or:

```
aws emr describe-cluster --cluster-id j-2AL4XXXXXX5T9
```

The output lists the cluster instances including DNS names and IP addresses. Note the value for `PublicDnsName`.

```
"Status": {
  "Timeline": {
    "ReadyDateTime": 1408040779.263,
    "CreationDateTime": 1408040515.535
  },
  "State": "RUNNING",
  "StateChangeReason": {}
},
"Ec2InstanceId": "i-e89b45e7",
"PublicDnsName": "ec2-###-##-##-###.us-west-2.compute.amazonaws.com"

"PrivateDnsName": "ip-###-##-##-###.us-west-2.compute.internal",
"PublicIpAddress": "##.###.###.##",
"Id": "ci-12XXXXXXXXFMH",
"PrivateIpAddress": "###.##.#.###"
```

For more information, see [Amazon EMR commands in the AWS CLI](#).

Connect to the Master Node Using SSH on Linux, Unix, and Mac OS X

Your Linux computer most likely includes an SSH client by default. For example, OpenSSH is installed on most Linux, Unix, and Mac OS X operating systems. You can check for an SSH client by typing `ssh` at the command line. If your computer doesn't recognize the command, you must install an SSH client to connect to the master node. The OpenSSH project provides a free implementation of the full suite of SSH tools. For more information, see the [OpenSSH](#) website.

The following instructions demonstrate opening an SSH connection to the Amazon EMR master node on Linux, Unix, and Mac OS X.

To configure the key pair private key file permissions

Before you can use your Amazon EC2 key pair private key to create an SSH connection, you must set permissions on the `.pem` file so that only the key owner has permission to access the file. This is required for creating an SSH connection using terminal or the AWS CLI.

1. Locate your `.pem` file. These instructions assume that the file is named `mykeypair.pem` and that it is stored in the current user's home directory.
2. Type the following command to set the permissions. Replace `~/mykeypair.pem` with the location and file name of your key pair private key file.

```
chmod 400 ~/mykeypair.pem
```

If you do not set permissions on the `.pem` file, you will receive an error indicating that your key file is unprotected and the key will be rejected. To connect, you only need to set permissions on the key pair private key file the first time you use it.

To connect to the master node using the terminal

1. Open a terminal window. On Mac OS X, choose **Applications > Utilities > Terminal**. On other Linux distributions, terminal is typically found at **Applications > Accessories > Terminal**.
2. To establish a connection to the master node, type the following command. Replace `ec2-###-###-###.compute-1.amazonaws.com` with the master public DNS name of your cluster and replace `~/mykeypair.pem` with the location and file name of your `.pem` file.

```
ssh hadoop@ec2-###-###-###.compute-1.amazonaws.com -i ~/mykeypair.pem
```

Important

You must use the login name `hadoop` when you connect to the Amazon EMR master node; otherwise, you may see an error similar to `Server refused our key`.

3. A warning states that the authenticity of the host you are connecting to cannot be verified. Type `yes` to continue.
4. When you are done working on the master node, type the following command to close the SSH connection.

```
exit
```

Connect to the Master Node Using the AWS CLI

You can create an SSH connection with the master node using the AWS CLI on Windows and on Linux, Unix, and Mac OS X. Regardless of the platform, you need the public DNS name of the master node and your Amazon EC2 key pair private key. If you are using the AWS CLI on Linux, Unix, or Mac OS X, you must also set permissions on the private key (`.pem` or `.ppk`) file as shown in [To configure the key pair private key file permissions \(p. 175\)](#).

To connect to the master node using the AWS CLI

1. To retrieve the cluster identifier, type:

```
aws emr list-clusters
```

The output lists your clusters including the cluster IDs. Note the cluster ID for the cluster to which you are connecting.

```
"Status": {
  "Timeline": {
    "ReadyDateTime": 1408040782.374,
    "CreationDateTime": 1408040501.213
  },
  "State": "WAITING",
  "StateChangeReason": {
    "Message": "Waiting after step completed"
  }
}
```

```
},  
"NormalizedInstanceHours": 4,  
"Id": "j-2AL4XXXXXX5T9",  
"Name": "AWS CLI cluster"
```

2. Type the following command to open an SSH connection to the master node. In the following example, replace `j-2AL4XXXXXX5T9` with the cluster ID and replace `~/mykeypair.key` with the location and file name of your `.pem` file (for Linux, Unix, and Mac OS X) or `.ppk` file (for Windows).

```
aws emr ssh --cluster-id j-2AL4XXXXXX5T9 --key-pair-file ~/mykeypair.key
```

3. When you are done working on the master node, close the AWS CLI window.

For more information, see [Amazon EMR commands in the AWS CLI](#).

Connect to the Master Node Using SSH on Windows

Windows users can use an SSH client such as PuTTY to connect to the master node. Before connecting to the Amazon EMR master node, you should download and install PuTTY and PuTTYgen. You can download these tools from the [PuTTY download page](#).

PuTTY does not natively support the key pair private key file format (`.pem`) generated by Amazon EC2. You use PuTTYgen to convert your key file to the required PuTTY format (`.ppk`). You must convert your key into this format (`.ppk`) before attempting to connect to the master node using PuTTY.

For more information about converting your key, see [Converting Your Private Key Using PuTTYgen](#) in the *Amazon EC2 User Guide for Linux Instances*.

To connect to the master node using PuTTY

1. Open `putty.exe`. You can also launch PuTTY from the Windows programs list.
2. If necessary, in the **Category** list, choose **Session**.
3. For **Host Name (or IP address)**, type `hadoop@MasterPublicDNS`. For example: `hadoop@ec2-###-##-##-###.compute-1.amazonaws.com`.
4. In the **Category** list, choose **Connection > SSH, Auth**.
5. For **Private key file for authentication**, choose **Browse** and select the `.ppk` file that you generated.
6. Choose **Open** and then **Yes** to dismiss the PuTTY security alert.

Important

When logging into the master node, type `hadoop` if you are prompted for a user name .

7. When you are done working on the master node, you can close the SSH connection by closing PuTTY.

Note

To prevent the SSH connection from timing out, you can choose **Connection** in the **Category** list and select the option **Enable TCP_keepalives**. If you have an active SSH session in PuTTY, you can change your settings by opening the context (right-click) for the PuTTY title bar and choosing **Change Settings**.

View Web Interfaces Hosted on Amazon EMR Clusters

Hadoop and other applications you install on your Amazon EMR cluster, publish user interfaces as web sites hosted on the master node. For security reasons, these web sites are only available on the master node's local web server and are not publicly available over the Internet. Hadoop also publishes user

interfaces as web sites hosted on the core and task (slave) nodes. These web sites are also only available on local web servers on the nodes.

The following table lists web interfaces that you can view on the core and task nodes. These Hadoop interfaces are available on all clusters. To access the following interfaces, replace `slave-public-dns-name` in the URI with the public DNS name of the node. For more information about retrieving the public DNS name of a core or task node instance, see [Connecting to Your Linux/Unix Instances Using SSH](#) in the *Amazon EC2 User Guide for Linux Instances*. In addition to retrieving the public DNS name of the core or task node, you must also edit the ElasticMapReduce-slave security group to allow SSH access over TCP port 22. For more information about modifying security group rules, see [Adding Rules to a Security Group](#) in the *Amazon EC2 User Guide for Linux Instances*.

Name of interface	URI
YARN ResourceManager	<code>http://master-public-dns-name:8088/</code>
YARN NodeManager	<code>http://slave-public-dns-name:8042/</code>
Hadoop HDFS NameNode	<code>http://master-public-dns-name:50070/</code>
Hadoop HDFS DataNode	<code>http://slave-public-dns-name:50075/</code>
Spark HistoryServer	<code>http://master-public-dns-name:18080/</code>
Zeppelin	<code>http://master-public-dns-name:8890/</code>
Hue	<code>http://master-public-dns-name:8888/</code>
Ganglia	<code>http://master-public-dns-name/ganglia/</code>
HBase UI	<code>http://master-public-dns-name:16010/</code>

Because there are several application-specific interfaces available on the master node that are not available on the core and task nodes, the instructions in this document are specific to the Amazon EMR master node. Accessing the web interfaces on the core and task nodes can be done in the same manner as you would access the web interfaces on the master node.

There are several ways you can access the web interfaces on the master node. The easiest and quickest method is to use SSH to connect to the master node and use the text-based browser, Lynx, to view the web sites in your SSH client. However, Lynx is a text-based browser with a limited user interface that cannot display graphics. The following example shows how to open the Hadoop ResourceManager interface using Lynx (Lynx URLs are also provided when you log into the master node using SSH).

```
lynx http://ip-###-##-###.us-west-2.compute.internal:8088/
```

There are two remaining options for accessing web interfaces on the master node that provide full browser functionality. Choose one of the following:

- Option 1 (recommended for more technical users): Use an SSH client to connect to the master node, configure SSH tunneling with local port forwarding, and use an Internet browser to open web interfaces hosted on the master node. This method allows you to configure web interface access without using a SOCKS proxy.
- Option 2 (recommended for new users): Use an SSH client to connect to the master node, configure SSH tunneling with dynamic port forwarding, and configure your Internet browser to use an add-on such as FoxyProxy or SwitchySharp to manage your SOCKS proxy settings. This method allows you to automatically filter URLs based on text patterns and to limit the proxy settings to domains that match the form of the master node's DNS name. The browser add-on automatically handles turning the proxy

on and off when you switch between viewing websites hosted on the master node, and those on the Internet. For more information about how to configure FoxyProxy for Firefox and Google Chrome, see [Option 2, Part 2: Configure Proxy Settings to View Websites Hosted on the Master Node \(p. 182\)](#).

Topics

- [Option 1: Set Up an SSH Tunnel to the Master Node Using Local Port Forwarding \(p. 179\)](#)
- [Option 2, Part 1: Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding \(p. 180\)](#)
- [Option 2, Part 2: Configure Proxy Settings to View Websites Hosted on the Master Node \(p. 182\)](#)
- [Access the Web Interfaces on the Master Node Using the Console \(p. 184\)](#)

Option 1: Set Up an SSH Tunnel to the Master Node Using Local Port Forwarding

To connect to the local web server on the master node, you create an SSH tunnel between your computer and the master node. This is also known as *port forwarding*. If you do not wish to use a SOCKS proxy, you can set up an SSH tunnel to the master node using local port forwarding. With local port forwarding, you specify unused local ports that are used to forward traffic to specific remote ports on the master node's local web server.

Setting up an SSH tunnel using local port forwarding requires the public DNS name of the master node and your key pair private key file. For information about how to locate the master public DNS name, see [To retrieve the public DNS name of the master node using the Amazon EMR console \(p. 174\)](#). For more information about accessing your key pair, see [Amazon EC2 Key Pairs](#) in the *Amazon EC2 User Guide for Linux Instances*. For more information about the sites you might want to view on the master node, see [View Web Interfaces Hosted on Amazon EMR Clusters \(p. 177\)](#).

Set Up an SSH Tunnel to the Master Node Using Local Port Forwarding on Linux, Unix, and Mac OS X

To set up an SSH tunnel using local port forwarding in terminal

1. Open a terminal window. On Mac OS X, choose **Applications > Utilities > Terminal**. On other Linux distributions, terminal is typically found at **Applications > Accessories > Terminal**.
2. Type the following command to open an SSH tunnel on your local machine. This command accesses the ResourceManager web interface by forwarding traffic on local port 8157 (a randomly chosen, unused local port) to port 80 on the master node's local web server. In the command, replace `~/mykeypair.pem` with the location and file name of your `.pem` file and replace `ec2-###-##-###-###.compute-1.amazonaws.com` with the master public DNS name of your cluster.

```
ssh -i ~/mykeypair.pem -N -L 8157:ec2-###-##-###-###.compute-1.amazonaws.com:8088  
hadoop@ec2-###-##-###-###.compute-1.amazonaws.com
```

After you issue this command, the terminal remains open and does not return a response.

Note

`-L` signifies the use of local port forwarding which allows you to specify a local port used to forward data to the identified remote port on the master node's local web server.

3. To open the ResourceManager web interface in your browser, type: `http://localhost:8157/` in the address bar.
4. When you are done working with the web interfaces on the master node, close the terminal windows.

Option 2, Part 1: Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding

To connect to the local web server on the master node, you create an SSH tunnel between your computer and the master node. This is also known as *port forwarding*. If you create your SSH tunnel using dynamic port forwarding, all traffic routed to a specified unused local port is forwarded to the local web server on the master node. This creates a SOCKS proxy. You can then configure your Internet browser to use an add-on such as FoxyProxy or SwitchySharp to manage your SOCKS proxy settings. Using a proxy management add-on allows you to automatically filter URLs based on text patterns and to limit the proxy settings to domains that match the form of the master node's public DNS name. The browser add-on automatically handles turning the proxy on and off when you switch between viewing websites hosted on the master node, and those on the Internet.

Before you begin, you need the public DNS name of the master node and your key pair private key file. For information about how to locate the master public DNS name, see [To retrieve the public DNS name of the master node using the Amazon EMR console \(p. 174\)](#). For more information about accessing your key pair, see [Amazon EC2 Key Pairs](#) in the *Amazon EC2 User Guide for Linux Instances*. For more information about the sites you might want to view on the master node, see [View Web Interfaces Hosted on Amazon EMR Clusters \(p. 177\)](#).

Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding on Linux, Unix, and Mac OS X

To set up an SSH tunnel using dynamic port forwarding on Linux, Unix, and Mac OS X

1. Open a terminal window. On Mac OS X, choose **Applications > Utilities > Terminal**. On other Linux distributions, terminal is typically found at **Applications > Accessories > Terminal**.
2. Type the following command to open an SSH tunnel on your local machine. Replace `~/mykeypair.pem` with the location and file name of your `.pem` file, replace `8157` with an unused, local port number, and replace `c2-###-##-##-###.compute-1.amazonaws.com` with the master public DNS name of your cluster.

```
ssh -i ~/mykeypair.pem -N -D 8157 hadoop@ec2-###-##-##-###.compute-1.amazonaws.com
```

After you issue this command, the terminal remains open and does not return a response.

Note

`-D` signifies the use of dynamic port forwarding which allows you to specify a local port used to forward data to all remote ports on the master node's local web server. Dynamic port forwarding creates a local SOCKS proxy listening on the port specified in the command.

3. After the tunnel is active, configure a SOCKS proxy for your browser. For more information, see [Option 2, Part 2: Configure Proxy Settings to View Websites Hosted on the Master Node \(p. 182\)](#).
4. When you are done working with the web interfaces on the master node, close the terminal window.

Set Up an SSH tunnel Using Dynamic Port Forwarding with the AWS CLI

You can create an SSH connection with the master node using the AWS CLI on Windows and on Linux, Unix, and Mac OS X. If you are using the AWS CLI on Linux, Unix, or Mac OS X, you must set permissions on the `.pem` file as shown in [To configure the key pair private key file permissions \(p. 175\)](#). If you are using the AWS CLI on Windows, PuTTY must appear in the path environment variable or you may receive an error such as OpenSSH or PuTTY not available.

To set up an SSH tunnel using dynamic port forwarding with the AWS CLI

1. Create an SSH connection with the master node as shown in [Connect to the Master Node Using the AWS CLI \(p. 176\)](#).
2. To retrieve the cluster identifier, type:

```
aws emr list-clusters
```

The output lists your clusters including the cluster IDs. Note the cluster ID for the cluster to which you are connecting.

```
"Status": {
  "Timeline": {
    "ReadyDateTime": 1408040782.374,
    "CreationDateTime": 1408040501.213
  },
  "State": "WAITING",
  "StateChangeReason": {
    "Message": "Waiting after step completed"
  }
},
"NormalizedInstanceHours": 4,
"Id": "j-2AL4XXXXXX5T9",
"Name": "AWS CLI cluster"
```

3. Type the following command to open an SSH tunnel to the master node using dynamic port forwarding. In the following example, replace `j-2AL4XXXXXX5T9` with the cluster ID and replace `~/mykeypair.key` with the location and file name of your `.pem` file (for Linux, Unix, and Mac OS X) or `.ppk` file (for Windows).

```
aws emr socks --cluster-id j-2AL4XXXXXX5T9 --key-pair-file ~/mykeypair.key
```

Note

The `socks` command automatically configures dynamic port forwarding on local port 8157. Currently, this setting cannot be modified.

4. After the tunnel is active, configure a SOCKS proxy for your browser. For more information, see [Option 2, Part 2: Configure Proxy Settings to View Websites Hosted on the Master Node \(p. 182\)](#).
5. When you are done working with the web interfaces on the master node, close the AWS CLI window.

For more information on using Amazon EMR commands in the AWS CLI, see <http://docs.aws.amazon.com/cli/latest/reference/emr>.

Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding on Windows

Windows users can use an SSH client such as PuTTY to create an SSH tunnel to the master node. Before connecting to the Amazon EMR master node, you should download and install PuTTY and PuTTYgen. You can download these tools from the [PuTTY download page](#).

PuTTY does not natively support the key pair private key file format (`.pem`) generated by Amazon EC2. You use PuTTYgen to convert your key file to the required PuTTY format (`.ppk`). You must convert your key into this format (`.ppk`) before attempting to connect to the master node using PuTTY.

For more information about converting your key, see [Converting Your Private Key Using PuTTYgen](#) in the *Amazon EC2 User Guide for Linux Instances*.

To set up an SSH tunnel using dynamic port forwarding on Windows

1. Double-click `putty.exe` to start PuTTY. You can also launch PuTTY from the Windows programs list.

Note

If you already have an active SSH session with the master node, you can add a tunnel by right-clicking the PuTTY title bar and choosing **Change Settings**.

2. If necessary, in the **Category** list, choose **Session**.
3. In the **Host Name** field, type `hadoop@MasterPublicDNS`. For example: `hadoop@ec2-###-##-##-###.compute-1.amazonaws.com`.
4. In the **Category** list, expand **Connection > SSH**, and then choose **Auth**.
5. For **Private key file for authentication**, choose **Browse** and select the `.ppk` file that you generated.

Note

PuTTY does not natively support the key pair private key file format (`.pem`) generated by Amazon EC2. You use PuTTYgen to convert your key file to the required PuTTY format (`.ppk`). You must convert your key into this format (`.ppk`) before attempting to connect to the master node using PuTTY.

6. In the **Category** list, expand **Connection > SSH**, and then choose **Tunnels**.
7. In the **Source port** field, type `8157` (an unused local port).
8. Leave the **Destination** field blank.
9. Select the **Dynamic** and **Auto** options.
10. Choose **Add** and **Open**.
11. Choose **Yes** to dismiss the PuTTY security alert.

Important

When you log in to the master node, type `hadoop` if you are prompted for a user name.

12. After the tunnel is active, configure a SOCKS proxy for your browser. For more information, see [Option 2, Part 2: Configure Proxy Settings to View Websites Hosted on the Master Node \(p. 182\)](#).
13. When you are done working with the web interfaces on the master node, close the PuTTY window.

Option 2, Part 2: Configure Proxy Settings to View Websites Hosted on the Master Node

If you use an SSH tunnel with dynamic port forwarding, you must use a SOCKS proxy management add-on to control the proxy settings in your browser. Using a SOCKS proxy management tool allows you to automatically filter URLs based on text patterns and to limit the proxy settings to domains that match the form of the master node's public DNS name. The browser add-on automatically handles turning the proxy on and off when you switch between viewing websites hosted on the master node and those on the Internet. To manage your proxy settings, configure your browser to use an add-on such as FoxyProxy or SwitchySharp.

For more information about creating an SSH tunnel, see [Option 2, Part 1: Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding \(p. 180\)](#). For more information about the available web interfaces, see [View Web Interfaces Hosted on Amazon EMR Clusters \(p. 177\)](#).

The following example demonstrates a FoxyProxy configuration using Google Chrome. The relevant settings that are loaded from the configuration file in the example are as follows:

- **Host or IP Address**—This is set to **localhost** with the Port set to **8157**. You should set this port should to the local port number that you used to establish the SSH tunnel with the master node in [Option 2, Part 1: Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding \(p. 180\)](#). This port must also match the port number you use in PuTTY or other terminal emulator you use to connect.
- **SOCKS v5** configuration is specified.

- Login credentials are not specified.
- **URL Patterns**

The following URL patterns are whitelisted and specified with a wildcard pattern type:

- The ***ec2*.amazonaws.com*** and ***10*.amazonaws.com*** patterns match the public DNS name of clusters in US regions.
- The ***ec2*.compute*** and ***10*.compute*** patterns match the public DNS name of clusters in all other regions.
- The **10.*** pattern provides access to the JobTracker log files in Hadoop. Alter this filter if it conflicts with your network access plan.

Configure FoxyProxy for Google Chrome

You can configure FoxyProxy for Google Chrome, Mozilla Firefox, and Microsoft Internet Explorer. FoxyProxy provides a set of proxy management tools that allow you to use a proxy server for URLs that match patterns corresponding to the domains used by the Amazon EC2 instances in your Amazon EMR cluster.

To install and configure FoxyProxy using Google Chrome

1. See <https://chrome.google.com/webstore/search/foxy%20proxy> and follow the links and instructions to add FoxyProxy to Chrome.
2. Using a text editor, create a file named `foxyproxy-settings.xml` with the following contents:

```
<?xml version="1.0" encoding="UTF-8"?>
<foxyproxy>
  <proxies>
    <proxy name="emr-socks-proxy" id="2322596116" notes="" fromSubscription="false"
      enabled="true" mode="manual" selectedTabIndex="2" lastresort="false"
      animatedIcons="true" includeInCycle="true" color="#0055E5" proxyDNS="true"
      noInternalIPs="false" autoconfMode="pac" clearCacheBeforeUse="false"
      disableCache="false" clearCookiesBeforeUse="false" rejectCookies="false">
      <matches>
        <match enabled="true" name="*ec2*.amazonaws.com*"
          pattern="*ec2*.amazonaws.com*" isRegex="false" isBlackList="false" isMultiLine="false"
          caseSensitive="false" fromSubscription="false" />
        <match enabled="true" name="*ec2*.compute*" pattern="*ec2*.compute*"
          isRegex="false" isBlackList="false" isMultiLine="false" caseSensitive="false"
          fromSubscription="false" />
        <match enabled="true" name="10.*" pattern="http://10.*"
          isRegex="false" isBlackList="false" isMultiLine="false" caseSensitive="false"
          fromSubscription="false" />
        <match enabled="true" name="*10*.amazonaws.com*"
          pattern="*10*.amazonaws.com*" isRegex="false" isBlackList="false" isMultiLine="false"
          caseSensitive="false" fromSubscription="false" />
        <match enabled="true" name="*10*.compute*" pattern="*10*.compute*"
          isRegex="false" isBlackList="false" isMultiLine="false" caseSensitive="false"
          fromSubscription="false" />
        <match enabled="true" name="*.compute.internal*"
          pattern="*.compute.internal*" isRegex="false" isBlackList="false" isMultiLine="false"
          caseSensitive="false" fromSubscription="false"/>
        <match enabled="true" name="*.ec2.internal*" pattern="*.ec2.internal*"
          isRegex="false" isBlackList="false" isMultiLine="false" caseSensitive="false"
          fromSubscription="false"/>
      </matches>
      <manualconf host="localhost" port="8157" socksVersion="5" isSocks="true"
        username="" password="" domain="" />
    </proxy>
  </proxies>
```

```
</foxyproxy>
```

3. Manage extensions in Chrome (go to <chrome://extensions>).
4. Choose **Options** for FoxyProxy Standard.
5. On the **FoxyProxy** page, choose **Import/Export**.
6. On the **Import/Export** page, choose **Choose File**, browse to the location of the `foxyproxy-settings.xml` file you created, select the file, and choose **Open**.
7. Choose **Replace** when prompted to overwrite the existing settings.
8. For **Proxy mode**, choose **Use proxies based on their predefined patterns and priorities**.
9. To open the web interfaces, in your browser's address bar, type `master-public-dns` followed by the port number or URL.

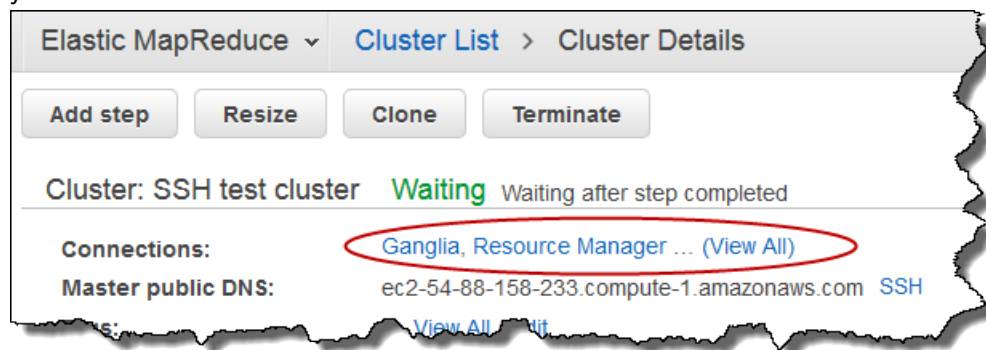
For a complete list of web interfaces on the master node, see [View Web Interfaces Hosted on Amazon EMR Clusters](#) (p. 177).

Access the Web Interfaces on the Master Node Using the Console

If you already have an SSH tunnel configured with the Amazon EMR master node using dynamic port forwarding, you can open the web interfaces using the console.

To open the web interfaces using the console

1. Verify that you have established an SSH tunnel with the master node and that you have a proxy management add-on configured for your browser.
2. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
3. On the **Cluster List** page, choose the link for your cluster.
4. In the cluster details, for **Connections**, choose the link for the web interface you wish to open in your browser.



5. Alternatively, choose the **View All** link to display links to all of the available web interfaces on your cluster's master node. Choosing the links opens the interfaces in your browser.

Web Interfaces Hosted on this Cluster

Hadoop, Ganglia, and other applications publish user interfaces as websites hosted on the master node. For security reasons, they are only available on the master node's local webserver (<http://localhost:port>) and are not published on the Internet.

Note

For the below links to work properly an SSH tunnel must be open and your browser configured to use the proxy for Amazon EMR.

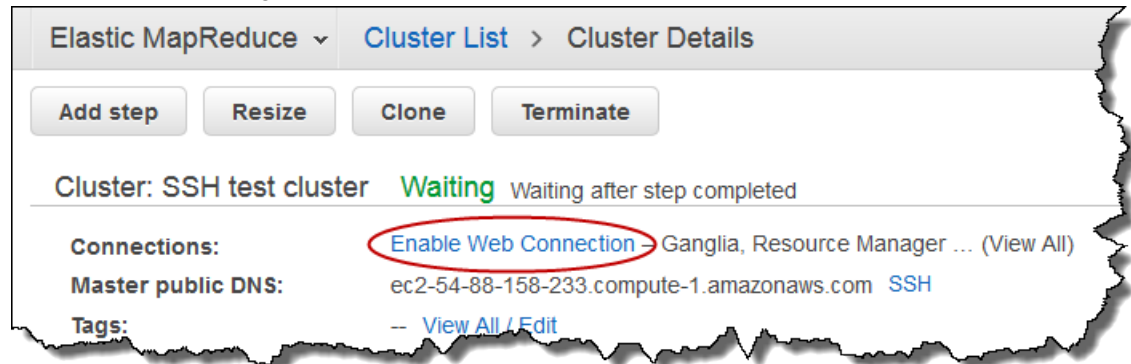
The following table lists web interfaces you can view on the master node:

Interface	URI
Resource Manager	http://ec2-54-164-54-29.compute-1.amazonaws.com:9026/
HDFS Name Node	http://ec2-54-164-54-29.compute-1.amazonaws.com:9101/

The following table lists web interfaces you can view on the slave nodes:

Interface	URI
Node Manager	http://ec2-000-000-000-000.compute-1.amazonaws.com:9035/
HDFS Data Node	http://ec2-000-000-000-000.compute-1.amazonaws.com:9102/

If you do not have an SSH tunnel open with the master node, choose **Enable Web Connection** for instructions on creating a tunnel.



Note

If you have an SSH tunnel configured using local port forwarding, the Amazon EMR console does not detect the connection.

Control Cluster Termination

Control over cluster termination is determined by two options: termination protection and auto-termination. By default, when you launch a cluster using the console, termination protection is turned on. This prevents accidental termination of the cluster. When you launch a cluster using the CLI or API, termination protection is turned off.

Auto-termination determines whether the cluster should automatically terminate when all steps are complete. When you launch a cluster using the console, the default behavior is for the cluster to remain active after all steps are complete. In other words, the cluster is long-running. A long-running cluster

must be manually terminated. When you launch a cluster using the CLI or API, the default behavior is for the cluster to terminate when data processing is complete; that is, when no more steps are left to run. This creates a transient cluster.

Topics

- [Terminate a Cluster \(p. 186\)](#)
- [Managing Cluster Termination \(p. 188\)](#)

Terminate a Cluster

This section describes the methods of terminating a cluster. You can terminate clusters in the `STARTING`, `RUNNING`, or `WAITING` states. A cluster in the `WAITING` state must be terminated or it runs indefinitely, generating charges to your account. You can terminate a cluster that fails to leave the `STARTING` state or is unable to complete a step.

If you are terminating a cluster that has termination protection set on it, you must first disable termination protection before you can terminate the cluster. After termination protection is disabled, you can terminate the cluster. Clusters can be terminated using the console, the AWS CLI, or programmatically using the `TerminateJobFlows` API.

Depending on the configuration of the cluster, it may take up to 5-20 minutes for the cluster to completely terminate and release allocated resources, such as EC2 instances.

Terminate a Cluster Using the Console

You can terminate one or more clusters using the Amazon EMR console. The steps to terminate a cluster in the console vary depending on whether termination protection is on or off. To terminate a protected cluster, you must first disable termination protection.

To terminate a cluster with termination protection off

1. Sign in to the AWS Management Console and open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Select the cluster to terminate. You can select multiple clusters and terminate them at the same time.
3. Choose **Terminate**.
4. When prompted, choose **Terminate**.

Amazon EMR terminates the instances in the cluster and stops saving log data.

To terminate a cluster with termination protection on

1. Sign in to the AWS Management Console and open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. On the **Cluster List** page, select the cluster to terminate. You can select multiple clusters and terminate them at the same time.
3. Choose **Terminate**.
4. When prompted, choose **Change** to turn termination protection off. If you selected multiple clusters, choose **Turn off all** to disable termination protection for all the clusters at once.
5. In the **Terminate clusters** dialog, for **Termination Protection**, choose **Off** and then click the check mark to confirm.
6. Click **Terminate**.

Amazon EMR terminates the instances in the cluster and stops saving log data.

Terminate a Cluster Using the AWS CLI

To terminate an unprotected cluster using the AWS CLI

To terminate an unprotected cluster using the AWS CLI, use the `terminate-clusters` subcommand with the `--cluster-ids` parameter.

- Type the following command to terminate a single cluster and replace `j-3KVXXXXXXXX7UG` with your cluster ID.

```
aws emr terminate-clusters --cluster-ids j-3KVXXXXXXXX7UG
```

To terminate multiple clusters, type the following command and replace `j-3KVXXXXXXXX7UG` and `j-WJ2XXXXXXXX8EU` with your cluster IDs.

```
aws emr terminate-clusters --cluster-ids j-3KVXXXXXXXX7UG j-WJ2XXXXXXXX8EU
```

For more information on using Amazon EMR commands in the AWS CLI, see <http://docs.aws.amazon.com/cli/latest/reference/emr>.

To terminate a protected cluster using the AWS CLI

To terminate a protected cluster using the AWS CLI, first disable termination protection using the `modify-cluster-attributes` subcommand with the `--no-termination-protected` parameter. Then use the `terminate-clusters` subcommand with the `--cluster-ids` parameter to terminate it.

- Type the following command to disable termination protection and replace `j-3KVTXXXXXXXX7UG` with your cluster ID.

```
aws emr modify-cluster-attributes --cluster-id j-3KVTXXXXXXXX7UG --no-termination-protected
```

- To terminate the cluster, type the following command and replace `j-3KVXXXXXXXX7UG` with your cluster ID.

```
aws emr terminate-clusters --cluster-ids j-3KVXXXXXXXX7UG
```

To terminate multiple clusters, type the following command and replace `j-3KVXXXXXXXX7UG` and `j-WJ2XXXXXXXX8EU` with your cluster IDs.

```
aws emr terminate-clusters --cluster-ids j-3KVXXXXXXXX7UG j-WJ2XXXXXXXX8EU
```

For more information on using Amazon EMR commands in the AWS CLI, see <http://docs.aws.amazon.com/cli/latest/reference/emr>.

Terminate a Cluster Using the API

The `TerminateJobFlows` operation ends step processing, uploads any log data from Amazon EC2 to Amazon S3 (if configured), and terminates the Hadoop cluster. A cluster also terminates automatically if you set `KeepJobAliveWhenNoSteps` to `False` in a `RunJobFlows` request.

You can use this action to terminate either a single cluster or a list of clusters by their cluster IDs.

For more information about the input parameters unique to `TerminateJobFlows`, see [TerminateJobFlows](#). For more information about the generic parameters in the request, see [Common Request Parameters](#).

Managing Cluster Termination

Termination protection ensures that the EC2 instances in your job flow are not shut down by an accident or error. This protection is especially useful if your cluster contains data in instance storage that you need to recover before those instances are terminated. When termination protection is not enabled, you can terminate clusters either through calls to the `TerminateJobFlows` API, through the Amazon EMR console, or by using the command line interface. In addition, the master node may terminate a node that has become unresponsive or has returned an error.

By default, termination protection is enabled when you launch a cluster using the console. Termination protection is disabled by default when you launch a cluster using the CLI or API. When termination protection is enabled, you must explicitly remove termination protection from the cluster before you can terminate it. With termination protection enabled, `TerminateJobFlows` cannot terminate the cluster and users cannot terminate the cluster using the CLI. Users terminating the cluster using the Amazon EMR console receive an extra confirmation box asking if they want to remove termination protection before terminating the cluster.

If you attempt to terminate a protected cluster with the API or CLI, the API returns an error, and the CLI exits with a non-zero return code.

When you submit steps to a cluster, the `ActionOnFailure` setting determines what the cluster does in response to any errors. The possible values for this setting are:

- `TERMINATE_JOB_FLOW`: If the step fails, terminate the cluster. If the cluster has termination protection enabled AND auto-terminate disabled, it will not terminate.
- `CANCEL_AND_WAIT`: If the step fails, cancel the remaining steps. If the cluster has auto-terminate disabled, the cluster will not terminate.
- `CONTINUE`: If the step fails, continue to the next step.

Termination Protection in Amazon EMR and Amazon EC2

An EMR cluster with termination protection enabled has the `disableAPITermination` attribute set for all EC2 instances in the cluster. If there is a conflict between the termination protection setting in Amazon EC2 and the setting in Amazon EMR for an instance, the Amazon EMR setting overrides the Amazon EC2 setting on the given instance. For example, if you use the Amazon EC2 console to *enable* termination protection on an EC2 instance in an Amazon EMR cluster that has termination protection *disabled*, Amazon EMR turns off termination protection on that EC2 instance and shuts down the instance when the rest of the cluster terminates.

Termination Protection and Spot Instances

Amazon EMR termination protection does not prevent an Amazon EC2 Spot Instance from terminating when the Spot Price rises above the maximum bid price.

Termination Protection and Auto-terminate

Enabling auto-terminate creates a transient cluster. The cluster automatically terminates when the last step is successfully completed even if termination protection is enabled.

Disabling auto-terminate causes instances in a cluster to persist after steps have successfully completed, but still allows the cluster to be terminated by user action, by errors, and by calls to `TerminateJobFlows` (if termination protection is disabled).

Note

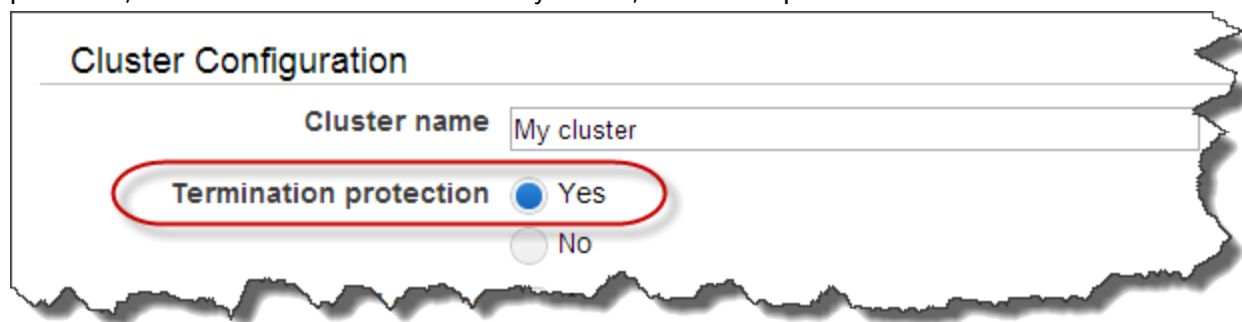
By default, auto-terminate is disabled for clusters launched using the console and the CLI. Clusters launched using the API have auto-terminate enabled.

Configuring Termination Protection for New Clusters

You can enable or disable termination protection when you launch a cluster using the console, the AWS CLI, or the API.

To configure termination protection for a new cluster using the console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**.
3. Choose **Go to advanced options**.
4. In the **Cluster Configuration** section, set the **Termination protection** field to **Yes** to enable protection, or set the field to **No** to disable it. By default, termination protection is enabled.



5. Proceed with creating the cluster.

To configure termination protection for a new cluster using the AWS CLI

Using the AWS CLI, you can launch a cluster with termination protection enabled by typing the `create-cluster` command with the `--termination-protected` parameter. By default, termination protection is disabled when you launch a cluster using the AWS CLI. You can also use the `--no-termination-protected` parameter to disable termination protection.

- To launch a protected cluster, type the following command and replace `myKey` with the name of your EC2 key pair.

```
aws emr create-cluster --name "Test cluster" --release-label emr-4.0.0 --applications
Name=Hadoop Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --
instance-type m3.xlarge --instance-count 3 --termination-protected
```

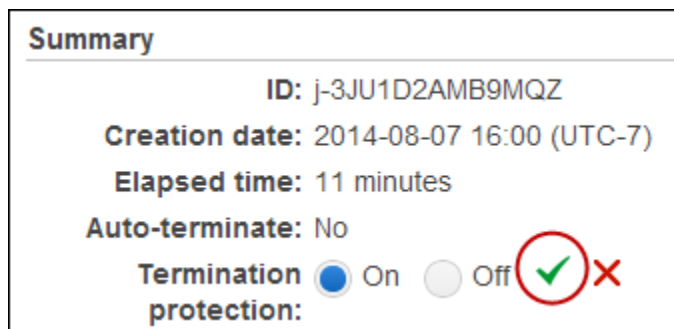
For more information on using Amazon EMR commands in the AWS CLI, see <http://docs.aws.amazon.com/cli/latest/reference/emr>.

Configuring Termination Protection for Running Clusters

You can configure termination protection for a running cluster using the console or the AWS CLI.

To configure termination protection for a running cluster using the console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. On the **Cluster List** page, click the link for your cluster.
3. On the **Cluster Details** page, in the **Summary** section, for **Termination protection**, click **Change**.
4. Click **On** and then click the check mark icon to enable termination protection. Alternatively, click **Off** to disable it.



To configure termination protection for a running cluster using the AWS CLI

To enable termination protection on a running cluster using the AWS CLI, type the `modify-cluster-attributes` subcommand with the `--termination-protected` parameter. To disable it, type the `--no-termination-protected` parameter.

- Type the following command to enable termination protection on a running cluster.

```
aws emr modify-cluster-attributes --cluster-id j-3KVTXXXXXX7UG --termination-protected
```

To disable termination protection, type:

```
aws emr modify-cluster-attributes --cluster-id j-3KVTXXXXXX7UG --no-termination-protected
```

Scaling Cluster Resources

You can adjust the number of Amazon EC2 instances available to an EMR cluster automatically or manually in response to workloads that have varying demands. The following options are available:

- You can configure *automatic scaling* for the core instance group and task instance groups when you first create them or after it's running. Amazon EMR automatically configures Auto Scaling parameters according to rules you specify, and then adds and removes instances based on a CloudWatch metric.
- You can manually *resize* the core instance group and task instance groups by manually adding or removing Amazon EC2 instances.
- You can add a new task instance group to the cluster.

The option to specify the Amazon EC2 instance type is only available during initial configuration of an instance group, so you can change the Amazon EC2 instance type only by adding a new task. A cluster-wide configuration allows you to specify whether Amazon EC2 instances removed from a cluster are terminated at the instance-hour boundary, or when tasks on the Amazon EC2 instance are complete. For more information, see [Configure Cluster Scale-Down \(p. 204\)](#).

Before you choose one of the methods for scaling described in this section, you should be familiar with some important concepts. First, you should understand the role of *node types* in an EMR cluster and how *instance groups* are used to manage them. For more information about the function of node types, see [What is Amazon EMR?](#), and for more information about instance groups, see [Instance Groups](#). You should also develop a strategy for right-sizing cluster resources based on the nature of your workload. For more information, see [Cluster Configuration Guidelines](#).

Note

The master instance group in an EMR cluster always consists of a single node running on a single Amazon EC2 instance, so it can't scale after you initially configure it. You work with the core instance groups and task instance groups to scale out and scale in a cluster. It's possible to have a cluster with only a master node, and no core or task nodes. You must have at least one core node at cluster creation in order to scale the cluster. In other words, single node clusters cannot be resized.

Topics

- [Using Automatic Scaling in Amazon EMR \(p. 191\)](#)
- [Manually Resizing a Running Cluster \(p. 199\)](#)
- [Configure Cluster Scale-Down \(p. 204\)](#)

Using Automatic Scaling in Amazon EMR

Automatic scaling in Amazon EMR allows you to programmatically scale out and scale in core nodes and task nodes in a cluster based on rules that you specify in a *scaling policy*. The scaling policy is part of an instance group configuration. You can specify a policy during initial configuration of an instance group, or by modifying an instance group in an existing cluster, even when that instance group is active. Each instance group in a cluster, except the master instance group, can have its own scaling policy, which consists of scale-out and scale-in rules. Scale-out and scale-in rules can be configured independently, with different parameters for each rule.

You can configure scaling policies using the AWS Management Console, the AWS CLI, or the Amazon EMR API. When you use the AWS CLI or Amazon EMR API, you specify the scaling policy in JSON format. When you initially create a scaling policy using the console, a default policy suitable for many applications is pre-configured to help you get started. You can delete or modify the default rules.

Even though automatic scaling allows you to adjust EMR cluster capacity on-the-fly, you should still consider baseline workload requirements and plan your node and instance group configurations. For more information, see [Cluster Configuration Guidelines](#).

Note

For most workloads, setting up both scale-in and scale-out rules is desirable to optimize resource utilization. Setting either rule without the other means that you need to manually resize the instance count after a scaling activity. In other words, this sets up a "one-way" automatic scale-out or scale-in policy with a manual reset.

Creating the IAM Role for Automatic Scaling

Automatic scaling in Amazon EMR requires an IAM role with permissions to add and terminate instances when scaling activities are triggered. A default role, `EMR_AutoScaling_DefaultRole` configured with the appropriate role policy and trust policy, is available for this purpose. Automatic scaling uses this IAM role to scale nodes on your behalf.

When you create a new cluster, you must create this role. Automatic scaling is not available on clusters that do not have this role, and the role cannot be added after the cluster is created. When you create a scaling policy using the AWS Management Console for Amazon EMR, the role is created by default. When you create a cluster that has an automatic scaling policy using the AWS CLI, you must use the `--auto-`

`scaling-role EMR_AutoScaling_DefaultRole` command to create the role. For more information, see [Creating a Cluster with an Automatic Scaling Policy Applied to an Instance Group \(p. 193\)](#).

Understanding Automatic Scaling Rules

When a scale-out rule triggers a scaling activity for an instance group, Amazon EC2 instances are added to the instance group according to your rules. New nodes can be used by applications such as Apache Spark and Apache Hive as soon as the Amazon EC2 instance enters the `InService` state. You can also set up a scale-in rule that terminates instances and removes nodes. For more information about the lifecycle of Amazon EC2 instances that scale automatically, see [Auto Scaling Lifecycle](#) in the *Auto Scaling User Guide*.

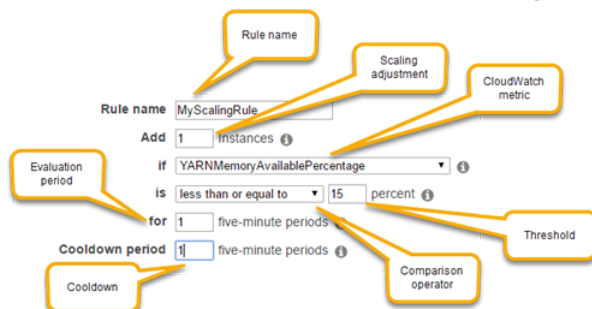
You can configure how a cluster terminates Amazon EC2 instances. You can choose to either terminate at the Amazon EC2 instance-hour boundary for billing, or upon task completion. This setting applies both to automatic scaling and to manual resizing operations. For more information about this configuration see [Configure Cluster Scale-Down \(p. 204\)](#).

The following parameters for each rule in a policy determine automatic scaling behavior.

Note

The parameters listed here are based on the AWS Management Console for Amazon EMR. When you use the AWS CLI or Amazon EMR API, additional advanced configuration options are available. For more information about advanced options, see [SimpleScalingPolicyConfiguration](#) in the *Amazon EMR API Reference*.

- Maximum instances and minimum instances. The **Maximum instances** constraint specifies the maximum number of Amazon EC2 instances that can be in the instance group, and applies to all scale-out rules. Similarly, the **Minimum instances** constraint specifies the minimum number of Amazon EC2 instances and applies to all scale-in rules.
- The **Rule name**, which must be unique within the policy.
- The **scaling adjustment**, which determines the number of EC2 instances to add (for scale-out rules) or terminate (for scale-in rules) during the scaling activity triggered by the rule.
- The **CloudWatch metric**, which is watched for an alarm condition.
- A **comparison operator**, which is used to compare the CloudWatch metric to the **Threshold** value and determine a trigger condition.
- An **evaluation period**, in five-minute increments, for which the CloudWatch metric must be in a trigger condition before scaling activity is triggered.
- A **Cooldown period**, which determines the amount of time that must elapse between a scaling activity started by a rule and the start of the next scaling activity, regardless of the rule that triggers it. When an instance group has finished a scaling activity and reached its post-scale state, the cooldown period provides an opportunity for the CloudWatch metrics that might trigger subsequent scaling activities to stabilize. For more information, see [Auto Scaling Cooldowns](#) in the *Auto Scaling User Guide*.



Using the AWS Management Console to Configure Automatic Scaling

When you create a cluster, you configure a scaling policy for instance groups using the advanced cluster configuration options. You can also create or modify a scaling policy for an instance group in-service by modifying instance groups in the **Hardware** settings of an existing cluster.

1. If you are creating a cluster, in the Amazon EMR console, select **Create Cluster**, select **Go to advanced options**, choose options for **Step 1: Software and Steps**, and then go to **Step 2: Hardware Configuration**.

—or—

If you are modifying an instance group in a running cluster, select your cluster from the cluster list, and then expand the **Hardware** section.

2. Click the pencil icon that appears in the **Auto Scaling** column for the instance group you want to configure. If an automatic scaling policy is already configured for the instance group, the number of **Maximum instances** and **Minimum instances** appear in this column; otherwise, **Not enabled** appears.

The Auto Scaling rules screen opens. **Scale out** and **Scale in** are selected by default, and default rules are pre-configured with settings suitable for many applications.

3. Type the **Maximum instances** you want the instance group to contain after it scales out, and type the **Minimum instances** you want the instance group to contain after it scales in.
4. Click the pencil to edit rule parameters, click the **X** to remove a rule from the policy, and click **Add rule** to add additional rules.
5. Choose rule parameters as described earlier in this topic. For descriptions of available CloudWatch metrics for Amazon EMR, see [Amazon EMR Metrics and Dimensions](#) in the *Amazon CloudWatch User Guide*.

Using the AWS CLI to Configure Automatic Scaling

You can use AWS CLI commands for Amazon EMR to configure automatic scaling when you create a cluster and when you create an instance group. You can use a shorthand syntax, specifying the JSON configuration inline within the relevant commands, or you can reference a file containing the configuration JSON. You can also apply an automatic scaling policy to an existing instance group and remove an automatic scaling policy that was previously applied. In addition, you can retrieve details of a scaling policy configuration from a running cluster.

Important

When you create a cluster that has an automatic scaling policy, you must use the `--auto-scaling-role EMR_AutoScaling_DefaultRole` command. To implement automatic scaling on a cluster, the cluster must have the `EMR_AutoScaling_DefaultRole`. The role can only be added when the cluster is created, and cannot be added to an existing cluster.

For a detailed description of the parameters available when configuring an automatic scaling policy, see [PutAutoScalingPolicy](#) in *Amazon EMR API Reference*.

Creating a Cluster with an Automatic Scaling Policy Applied to an Instance Group

You can specify an automatic scaling configuration within the `--instance-groups` option of the `aws emr create-cluster` command. The following example illustrates a create-cluster command where an automatic scaling policy for the core instance group is provided inline. The command creates a scaling configuration equivalent to the default scale-out policy that appears when you create an auto scaling

policy using the AWS Management Console for Amazon EMR. For brevity, a scale-in policy is not shown. Creating a scale-out rule without a scale-in rule is not recommended.

```
aws emr create-cluster --release-label emr-5.2.0 --service-role
EMR_DefaultRole --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole
--auto-scaling-role EMR_AutoScaling_DefaultRole --instance-groups
Name=MyMasterIG,InstanceGroupType=MASTER,InstanceType=m3.xlarge,InstanceCount=1
'Name=MyCoreIG,InstanceGroupType=CORE,InstanceType=m3.xlarge,InstanceCount=2,AutoScalingPolicy={Constr
scale-out,Description=Replicates the default scale-out rule in the
console.,Action={SimpleScalingPolicyConfiguration={AdjustmentType=CHANGE_IN_CAPACITY,ScalingAdjustment
ElasticMapReduce,Period=300,Statistic=AVERAGE,Threshold=15,Unit=PERCENT,Dimensions=[{Key=JobFlowId,Valu
```

The following command illustrates using the command line to provide the automatic scaling policy definition as part of an instance group configuration file named *instancegroupconfig.json*.

```
aws emr create-cluster --release-label emr-5.2.0 --service-role EMR_DefaultRole --ec2-
attributes InstanceProfile=EMR_EC2_DefaultRole --instance-groups file://your/path/to/
instancegroupconfig.json --auto-scaling-role EMR_AutoScaling_DefaultRole
```

With the contents of the configuration file as follows:

```
[
{
  "InstanceCount": 1,
  "Name": "MyMasterIG",
  "InstanceGroupType": "MASTER",
  "InstanceType": "m3.xlarge"
},
{
  "InstanceCount": 2,
  "Name": "MyCoreIG",
  "InstanceGroupType": "CORE",
  "InstanceType": "m3.xlarge",
  "AutoScalingPolicy":
  {
    "Constraints":
    {
      "MinCapacity": 2,
      "MaxCapacity": 10
    },
    "Rules":
    [
      {
        "Name": "Default-scale-out",
        "Description": "Replicates the default scale-out rule in the console for YARN
memory.",
        "Action":{
          "SimpleScalingPolicyConfiguration":{
            "AdjustmentType": "CHANGE_IN_CAPACITY",
            "ScalingAdjustment": 1,
            "CoolDown": 300
          }
        },
        "Trigger":{
          "CloudWatchAlarmDefinition":{
            "ComparisonOperator": "LESS_THAN",
            "EvaluationPeriods": 1,
            "MetricName": "YARNMemoryAvailablePercentage",
```

```
    "Namespace": "AWS/ElasticMapReduce",
    "Period": 300,
    "Threshold": 15,
    "Statistic": "AVERAGE",
    "Unit": "PERCENT",
    "Dimensions": [
      {
        "Key": "JobFlowId",
        "Value": "${emr.clusterId}"
      }
    ]
  }
}
]
```

Adding an Instance Group with an Automatic Scaling Policy to a Cluster

You can specify a scaling policy configuration using the `--instance-groups` option with the `add-instance-groups` command in the same way you can when you use `create-cluster`. The following example uses a reference to a JSON file, `instancegroupconfig.json`, with the instance group configuration.

```
aws emr add-instance-groups --cluster-id j-1EKZ3TYEVF1S2 --instance-groups file://your/path/to/instancegroupconfig.json
```

Applying an Automatic Scaling Policy to an Existing Instance Group or Modifying an Applied Policy

Use the `aws emr put-auto-scaling-policy` command to apply an automatic scaling policy to an existing instance group. The instance group must be part of a cluster that uses the automatic scaling IAM role. The following example uses a reference to a JSON file, `autoscaleconfig.json`, that specifies the automatic scaling policy configuration.

```
aws emr put-auto-scaling-policy --cluster-id j-1EKZ3TYEVF1S2 --instance-group-id ig-3PLUZBA6WLS07 --auto-scaling-policy file://your/path/to/autoscaleconfig.json
```

The contents of the `autoscaleconfig.json` file, which defines the same scale-out rule as shown in the previous example, is shown below.

```
"AutoScalingPolicy":
{
  "Constraints":
  {
    "MinCapacity": 2,
    "MaxCapacity": 10
  },
  "Rules":
  [
    {
      "Name": "Default-scale-out",
      "Description": "Replicates the default scale-out rule in the console for YARN memory.",

```



```
"Action":{
  "SimpleScalingPolicyConfiguration":{
    "AdjustmentType": "CHANGE_IN_CAPACITY",
    "ScalingAdjustment": 1,
    "CoolDown": 300
  }
},
"Trigger":{
  "CloudWatchAlarmDefinition":{
    "ComparisonOperator": "LESS_THAN",
    "EvaluationPeriods": 1,
    "MetricName": "YARNMemoryAvailablePercentage",
    "Namespace": "AWS/ElasticMapReduce",
    "Period": 300,
    "Threshold": 15,
    "Statistic": "AVERAGE",
    "Unit": "PERCENT",
    "Dimensions":[
      {
        "Key" : "JobFlowId",
        "Value" : "${emr.clusterId}"
      }
    ]
  }
}
}
```

Removing an Automatic Scaling Policy from an Instance Group

```
aws emr remove-auto-scaling-policy --cluster-id j-1EKZ3TYEVF1S2 --instance-group-id ig-3PLUZBA6WLS07
```

Retrieving an Automatic Scaling Policy Configuration

The `describe-cluster` command retrieves the policy configuration in the InstanceGroup block. For example, the following command retrieves the configuration for the cluster with a cluster ID of `j-1CWOHP4PI30VJ`.

```
aws emr describe-cluster --cluster-id j-1CWOHP4PI30VJ
```

The command produces the following example output.

```
{
  "Cluster": {
    "Configurations": [],
    "Id": "j-1CWOHP4PI30VJ",
    "NormalizedInstanceHours": 48,
    "Name": "Auto Scaling Cluster",
    "ReleaseLabel": "emr-5.2.0",
    "ServiceRole": "EMR_DefaultRole",
    "AutoTerminate": false,
    "TerminationProtected": true,
    "MasterPublicDnsName": "ec2-54-167-31-38.compute-1.amazonaws.com",
    "LogUri": "s3n://aws-logs-232939870606-us-east-1/elasticmapreduce/"
```

```
"Ec2InstanceAttributes": {
  "Ec2KeyName": "performance",
  "AdditionalMasterSecurityGroups": [],
  "AdditionalSlaveSecurityGroups": [],
  "EmrManagedSlaveSecurityGroup": "sg-09fc9362",
  "Ec2AvailabilityZone": "us-east-1d",
  "EmrManagedMasterSecurityGroup": "sg-0bfc9360",
  "IamInstanceProfile": "EMR_EC2_DefaultRole"
},
"Applications": [
  {
    "Name": "Hadoop",
    "Version": "2.7.3"
  }
],
"InstanceGroups": [
  {
    "AutoScalingPolicy": {
      "Status": {
        "State": "ATTACHED",
        "StateChangeReason": {
          "Message": ""
        }
      },
    },
    "Constraints": {
      "MaxCapacity": 10,
      "MinCapacity": 2
    },
    "Rules": [
      {
        "Name": "Default-scale-out",
        "Trigger": {
          "CloudWatchAlarmDefinition": {
            "MetricName": "YARNMemoryAvailablePercentage",
            "Unit": "PERCENT",
            "Namespace": "AWS/ElasticMapReduce",
            "Threshold": 15,
            "Dimensions": [
              {
                "Key": "JobFlowId",
                "Value": "j-1CWOHP4PI30VJ"
              }
            ],
            "EvaluationPeriods": 1,
            "Period": 300,
            "ComparisonOperator": "LESS_THAN",
            "Statistic": "AVERAGE"
          },
        },
        "Description": "",
        "Action": {
          "SimpleScalingPolicyConfiguration": {
            "CoolDown": 300,
            "AdjustmentType": "CHANGE_IN_CAPACITY",
            "ScalingAdjustment": 1
          }
        }
      },
      {
        "Name": "Default-scale-in",
        "Trigger": {
          "CloudWatchAlarmDefinition": {
            "MetricName": "YARNMemoryAvailablePercentage",
            "Unit": "PERCENT",
            "Namespace": "AWS/ElasticMapReduce",
            "Threshold": 0.75,
```

```

        "Dimensions": [
            {
                "Key": "JobFlowId",
                "Value": "j-1CWOHP4PI30VJ"
            }
        ],
        "EvaluationPeriods": 1,
        "Period": 300,
        "ComparisonOperator": "GREATER_THAN",
        "Statistic": "AVERAGE"
    }
},
"Description": "",
"Action": {
    "SimpleScalingPolicyConfiguration": {
        "Cooldown": 300,
        "AdjustmentType": "CHANGE_IN_CAPACITY",
        "ScalingAdjustment": -1
    }
}
}
]
},
"Configurations": [],
"InstanceType": "m3.xlarge",
"Market": "ON_DEMAND",
"Name": "Core - 2",
"ShrinkPolicy": {},
"Status": {
    "Timeline": {
        "CreationDateTime": 1479413437.342,
        "ReadyDateTime": 1479413864.615
    },
    "State": "RUNNING",
    "StateChangeReason": {
        "Message": ""
    }
},
"RunningInstanceCount": 2,
"Id": "ig-3M16XBE8C3PH1",
"InstanceGroupType": "CORE",
"RequestedInstanceCount": 2,
"EbsBlockDevices": []
},
{
    "Configurations": [],
    "Id": "ig-OP62I28NSE8M",
    "InstanceGroupType": "MASTER",
    "InstanceType": "m3.xlarge",
    "Market": "ON_DEMAND",
    "Name": "Master - 1",
    "ShrinkPolicy": {},
    "EbsBlockDevices": [],
    "RequestedInstanceCount": 1,
    "Status": {
        "Timeline": {
            "CreationDateTime": 1479413437.342,
            "ReadyDateTime": 1479413752.088
        },
        "State": "RUNNING",
        "StateChangeReason": {
            "Message": ""
        }
    },
    "RunningInstanceCount": 1
}
}

```

```
    ],
    "AutoScalingRole": "EMR_AutoScaling_DefaultRole",
    "Tags": [],
    "VisibleToAllUsers": true,
    "BootstrapActions": [],
    "Status": {
      "Timeline": {
        "CreationDateTime": 1479413437.339,
        "ReadyDateTime": 1479413863.666
      },
      "State": "WAITING",
      "StateChangeReason": {
        "Message": "Cluster ready after last step completed."
      }
    }
  }
}
```

Manually Resizing a Running Cluster

You can resize the core instance group in a running cluster by adding Amazon EC2 instances using the AWS Management Console, AWS CLI, or the Amazon EMR API. Applications can use newly provisioned Amazon EC2 instance capacity to host nodes as soon as the new nodes become available. You can also shrink the size of the instance group. A cluster can either be configured to terminate Amazon EC2 instances at the instance-hour boundary (the default in clusters created with Amazon EMR version 5.1.0 and later), or to terminate Amazon EC2 instances only after all processes on the nodes have completed. For more information, see [Configure Cluster Scale-Down \(p. 204\)](#).

Amazon EMR withstands slave node failures and continues cluster execution even if a slave node becomes unavailable. Amazon EMR automatically provisions additional slave nodes to replace those that fail.

You can have a different number of slave nodes for each cluster step. You can also add a step to a running cluster to modify the number of slave nodes. Because all steps are guaranteed to run sequentially by default, you can specify the number of running slave nodes for any step.

Resize a Cluster Using the Console

You can use the Amazon EMR console to resize a running cluster.

To resize a running cluster using the console

1. From the **Cluster List** page, choose a cluster to resize.
2. On the **Cluster Details** page, choose **Resize**. Alternatively, you can expand the **Hardware Configuration** section, and choose the **Resize** button adjacent to the core or task groups.
3. To add nodes to the core group or to one or more task groups, choose the **Resize** link in the **Count** column, change the number of instances, and choose the green check mark.

To add additional task instance groups, choose **Add task instance group**, choose the task node type, the number of task nodes, and whether the task nodes are Spot instances. If you attempt to add more than 48 task groups, you receive an error message. If your cluster was launched without a task group, choose **Add task nodes** to add one.

When you make a change to the number of nodes, the Amazon EMR console updates the status of the instance group through the **Provisioning** and **Resizing** states until they are ready and indicate

in brackets the newly requested number of nodes. When the change to the node count finishes, the instance groups return to the **Running** state.

Resize a Cluster Using the AWS CLI

You can use the AWS CLI to resize a running cluster. You can increase or decrease the number of task nodes, and you can increase the number of core nodes in a running cluster. It is also possible to terminate an instance in the core instance group using the AWS CLI or the API. This should be done with caution. Terminating an instance in the core instance group risks data loss, and the instance is not automatically replaced.

In addition to resizing the core and task groups, you can also add one or more task instance groups to a running cluster using the AWS CLI.

To resize a cluster by changing the instance count using the AWS CLI

You can add instances to the core group or task group, and you can remove instances from the task group using the AWS CLI `modify-instance-groups` subcommand with the `InstanceCount` parameter. To add instances to the core or task groups, increase the `InstanceCount`. To reduce the number of instances in the task group, decrease the `InstanceCount`. Changing the instance count of the task group to 0 removes all instances but not the instance group.

- To increase the number of instances in the task instance group from 3 to 4, type the following command and replace `ig-31JXXXXXXBTO` with the instance group ID.

```
aws emr modify-instance-groups --instance-groups
  InstanceGroupId=ig-31JXXXXXXBTO, InstanceCount=4
```

To retrieve the `InstanceGroupId`, use the `describe-cluster` subcommand. The output is a JSON object called `Cluster` that contains the ID of each instance group. To use this command, you need the cluster ID (which you can retrieve using the `aws emr list-clusters` command or the console). To retrieve the instance group ID, type the following command and replace `j-2AXXXXXXGAPLF` with the cluster ID.

```
aws emr describe-cluster --cluster-id j-2AXXXXXXGAPLF
```

Using the AWS CLI, you can also terminate an instance in the core instance group with the `--modify-instance-groups` subcommand. This should be done with caution. Terminating an instance in the core instance group risks data loss, and the instance is not automatically replaced. To terminate a specific instance you need the instance group ID (returned by the `aws emr describe-cluster --cluster-id` subcommand) and the instance ID (returned by the `aws emr list-instances --cluster-id` subcommand), type the following command, replace `ig-6RXXXXXX07SA` with the instance group ID and replace `i-f9XXXXf2` with the instance ID.

```
aws emr modify-instance-groups --instance-groups
  InstanceGroupId=ig-6RXXXXXX07SA, EC2InstanceIdsToTerminate=i-f9XXXXf2
```

For more information on using Amazon EMR commands in the AWS CLI, see <http://docs.aws.amazon.com/cli/latest/reference/emr>.

To resize a cluster by adding task instance groups using the AWS CLI

Using the AWS CLI, you can add between one and 48 task instance groups to a cluster with the `--add-instance-groups` subcommand. Task instances groups can only be added to a cluster containing a master instance group and a core instance group. When using the AWS CLI, you can add up to 5 task instance groups each time you use the `--add-instance-groups` subcommand.

1. To add a single task instance group to a cluster, type the following command and replace `j-JXBXXXXXX37R` with the cluster ID.

```
aws emr add-instance-groups --cluster-id j-JXBXXXXXX37R --instance-groups
  InstanceCount=6,InstanceGroupType=task,InstanceType=m1.large
```

2. To add multiple task instance groups to a cluster, type the following command and replace `j-JXBXXXXXX37R` with the cluster ID. You can add up to 5 task instance groups in a single command.

```
aws emr add-instance-groups --cluster-id j-JXBXXXXXX37R --instance-
groups InstanceCount=6,InstanceGroupType=task,InstanceType=m1.large
  InstanceCount=10,InstanceGroupType=task,InstanceType=m3.xlarge
```

For more information on using Amazon EMR commands in the AWS CLI, see <http://docs.aws.amazon.com/cli/latest/reference/emr>.

Interrupting a Resize

Note

This feature is for Amazon EMR releases 4.1.0 or greater.

You can issue a resize in the midst of an existing resize operation. Additionally, you can stop a previously submitted resize request or submit a new request to override a previous request without waiting for it to finish. You can also stop an existing resize from the console or using the `ModifyInstanceGroups` API call with the current count as the target count of the cluster.

The following screenshot shows a task instance group that is resizing but can be stopped by choosing **Stop**.

Task instance group

To interrupt a resize using the CLI

You can use the AWS CLI to stop a resize by using the `modify-instance-groups` subcommand. Assume you have six instances in your instance group and you want to increase this to 10. You later decide that you would like to cancel this request:

- The initial request:

```
aws emr modify-instance-groups --instance-groups
  InstanceGroupId=ig-myInstanceGroupId,InstanceCount=10
```

The second request to stop the first request:

```
aws emr modify-instance-groups --instance-groups
  InstanceGroupId=ig-myInstanceGroupId,InstanceCount=6
```

Note

Because this process is asynchronous, you may see instance counts change with respect to previous API requests before subsequent requests are honored. In the case of shrinking, it is possible that if you have work running on the nodes, the instance group may not shrink until nodes have completed their work.

Arrested State

An instance group goes into arrested state if it encounters too many errors while trying to start the new cluster nodes. For example, if new nodes fail while performing bootstrap actions, the instance group goes into an *ARRESTED* state, rather than continuously provisioning new nodes. After you resolve the underlying issue, reset the desired number of nodes on the cluster's instance group, and then the instance group resumes allocating nodes. Modifying an instance group instructs Amazon EMR to attempt to provision nodes again. No running nodes are restarted or terminated.

In the AWS CLI, the `list-instances` subcommand returns all instances and their states as does the `describe-cluster` subcommand. If Amazon EMR detects a fault with an instance group, it changes the group's state to *ARRESTED*.

To reset a cluster in an *ARRESTED* state using the AWS CLI

Type the `describe-cluster` subcommand with the `--cluster-id` parameter to view the state of the instances in your cluster.

- To view information on all instances and instance groups in a cluster, type the following command and replace `j-3KVXXXXXX7UG` with the cluster ID.

```
aws emr describe-cluster --cluster-id j-3KVXXXXXX7UG
```

The output displays information about your instance groups and the state of the instances:

```
{
  "Cluster": {
    "Status": {
      "Timeline": {
        "ReadyDateTime": 1413187781.245,
        "CreationDateTime": 1413187405.356
      },
      "State": "WAITING",
      "StateChangeReason": {
        "Message": "Waiting after step completed"
      }
    },
    "Ec2InstanceAttributes": {
      "Ec2AvailabilityZone": "us-west-2b"
    },
    "Name": "Development Cluster",
    "Tags": [],
    "TerminationProtected": false,
    "RunningAmiVersion": "3.2.1",
    "NormalizedInstanceHours": 16,
    "InstanceGroups": [
      {
        "RequestedInstanceCount": 1,
        "Status": {
          "Timeline": {
            "ReadyDateTime": 1413187775.749,
            "CreationDateTime": 1413187405.357
          },
          "State": "RUNNING",
          "StateChangeReason": {
            "Message": ""
          }
        },
        "Name": "MASTER",
        "InstanceGroupType": "MASTER",
        "InstanceType": "m1.large",
```

```
    "Id": "ig-3ETXXXXXXFYV8",
    "Market": "ON_DEMAND",
    "RunningInstanceCount": 1
  },
  {
    "RequestedInstanceCount": 1,
    "Status": {
      "Timeline": {
        "ReadyDateTime": 1413187781.301,
        "CreationDateTime": 1413187405.357
      },
      "State": "RUNNING",
      "StateChangeReason": {
        "Message": ""
      }
    },
    "Name": "CORE",
    "InstanceGroupType": "CORE",
    "InstanceType": "m1.large",
    "Id": "ig-3SUXXXXXXQ9ZM",
    "Market": "ON_DEMAND",
    "RunningInstanceCount": 1
  }
  ...
}
```

To view information on a particular instance group, type the `list-instances` subcommand with the `--cluster-id` and `--instance-group-types` parameters. You can view information for the MASTER, CORE, or TASK groups.

```
aws emr list-instances --cluster-id j-3KVXXXXXXY7UG --instance-group-types "CORE"
```

Use the `modify-instance-groups` subcommand with the `--instance-groups` parameter to reset a cluster in the `ARRESTED` state. The instance group id is returned by the `describe-cluster` subcommand.

```
aws emr modify-instance-groups --instance-groups
  InstanceGroupId=ig-3SUXXXXXXQ9ZM, InstanceCount=3
```


Configure Cluster Scale-Down

With Amazon EMR versions 5.1.0 and later, you can configure the scale-down behavior of Amazon EC2 instances when a termination request is issued. A termination request can come from an automatic scaling policy that triggers a scale-in activity, or from the manual removal of instances from an instance group when a cluster is resized. There are two options for scale-down behavior: terminate at the instance-hour boundary for Amazon EC2 billing, or terminate at task completion. You can use the AWS Management Console for Amazon EMR, the AWS CLI, or the Amazon EMR API. Terminating at the instance-hour boundary is the default, regardless of when the request to terminate the instance was submitted. Because Amazon EC2 charges per full hour regardless of when the instance is terminated, this behavior enables applications running on your cluster to more cost-effectively utilize Amazon EC2 instances in a dynamically scaling environment. Amazon EMR terminates nodes with the least tasks or no tasks first.

For clusters created using Amazon EMR versions earlier than 5.1.0, and beginning with version 4.1.0, Amazon EMR terminates Amazon EC2 instances at task completion by default. Specifying termination at the instance-hour boundary is not available in these versions. When terminate at task completion is specified, Amazon EMR blacklists and drains tasks from nodes before terminating the Amazon EC2 instances, regardless of the instance-hour boundary.

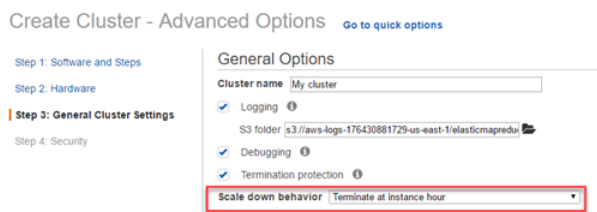
With either behavior Amazon EMR does not terminate EC2 instances in core instance groups if it could lead to HDFS corruption.

Configuring Amazon EMR Scale-Down Behavior

Note

This configuration feature is only available for Amazon EMR releases 5.1.0 or later.

You can use the AWS Management Console, the AWS CLI, or the Amazon EMR API to configure scale-down behavior when you create a cluster. Configuring scale-down using the AWS Management Console is done in the **Step 3: General Cluster Settings** screen when you create a cluster using **Advanced options**.



When you create a cluster using the AWS CLI, use the `--ScaleDownBehavior` option to specify either `TERMINATE_AT_INSTANCE_HOUR` or `TERMINATE_AT_TASK_COMPLETION`.

Terminate at Task Completion

Amazon EMR allows you to scale down your cluster without affecting your workload. Amazon EMR gracefully decommissions YARN, HDFS, and other daemons on core and task nodes during a resize down operation without losing data or interrupting jobs. Amazon EMR only shrinks instance groups if the work assigned to the groups has completed and they are idle. For YARN NodeManager decommissioning, you can manually adjust the time a node waits for decommissioning by setting `yarn.resourcemanager.decommissioning.timeout` inside `/etc/hadoop/conf/yarn-site.xml`; this setting is dynamically propagated. If there are still running containers or YARN applications when the decommissioning timeout passes, the node is forced to be decommissioned and YARN reschedules affected containers on other nodes. The default value is 3600s (one hour), meaning a YARN node under that has been issued a resize down request will become decommissioned within one hour or less. You can set this timeout to be an arbitrarily high value to force graceful shrink to wait longer.

Task Node Groups

Amazon EMR will intelligently select instances which are not running tasks and remove them from a cluster first. If all instances in the cluster are being utilized, Amazon EMR will wait for tasks to complete on a given instance before removing it from the cluster. The default wait time is 1 hour and this value can be changed by setting `yarn.resourcemanager.decommissioning.timeout`. Amazon EMR will dynamically use the new setting. You can set this to an arbitrarily large number to ensure that no tasks are killed while shrinking the cluster.

Core Node Groups

On core nodes, both YARN NodeManager and HDFS DataNode daemons must be decommissioned in order for the instance group to shrink. For YARN, graceful shrink ensures that a node marked for decommissioning is only transitioned to the DECOMMISSIONED state if there are no pending or incomplete containers or applications. The decommissioning finishes immediately if there are no running containers on the node at the beginning of decommissioning.

For HDFS, graceful shrink ensures that the target capacity of HDFS is large enough to fit all existing blocks. If the target capacity is not large enough, only a partial amount of core instances are decommissioned such that the remaining nodes can handle the current data residing in HDFS. You should ensure additional HDFS capacity to allow further decommissioning. You should also try to minimize write I/O before attempting to shrink instance groups as that may delay the completion of the resize operation.

Another limit is the default replication factor, `dfs.replication` inside `/etc/hadoop/conf/hdfs-site`. Amazon EMR configures the value based on the number of instances in the cluster: 1 with 1-3 instances, 2 for clusters with 4-9 instances, and 3 for clusters with 10+ instances. Graceful shrink does not allow you to shrink core nodes below the HDFS replication factor; this is to prevent HDFS from being unable to close files due insufficient replicas. To circumvent this limit, you must lower the replication factor and restart the NameNode daemon.

Cloning a Cluster Using the Console

You can use the Amazon EMR console to clone a cluster, which makes a copy of the configuration of the original cluster to use as the basis for a new cluster.

To clone a cluster using the console

1. From the **Cluster List** page, click a cluster to clone.
2. At the top of the **Cluster Details** page, click **Clone**.

In the dialog box, choose **Yes** to include the steps from the original cluster in the cloned cluster. Choose **No** to clone the original cluster's configuration without including any of the steps.

Note

For clusters created using AMI 3.1.1 and later (Hadoop 2.x) or AMI 2.4.8 and later (Hadoop 1.x), if you clone a cluster and include steps, all system steps (such as configuring Hive) are cloned along with user-submitted steps, up to 1,000 total. Any older steps that no longer appear in the console's step history cannot be cloned. For earlier AMIs, only 256 steps can be cloned (including system steps). For more information, see [Submit Work to a Cluster \(p. 206\)](#).

3. The **Create Cluster** page appears with a copy of the original cluster's configuration. Review the configuration, make any necessary changes, and then click **Create Cluster**.

Submit Work to a Cluster

This section describes the methods for submitting work to an Amazon EMR cluster. You can submit work to a cluster by adding steps or by interactively submitting Hadoop jobs to the master node. The maximum number of PENDING and ACTIVE steps allowed in a cluster is 256. You can submit jobs interactively to the master node even if you have 256 active steps running on the cluster. You can submit an unlimited number of steps over the lifetime of a long-running cluster, but only 256 steps can be ACTIVE or PENDING at any given time.

Topics

- [Work with Steps Using the CLI and Console \(p. 206\)](#)
- [Submit Hadoop Jobs Interactively \(p. 208\)](#)
- [Add More than 256 Steps to a Cluster \(p. 210\)](#)

Work with Steps Using the CLI and Console

You can add steps to a cluster using the AWS Management Console, the AWS CLI, or the Amazon EMR API. The maximum number of PENDING and ACTIVE steps allowed in a cluster is 256, which includes system steps such as install Pig, install Hive, install HBase, and configure debugging. You can submit an unlimited number of steps over the lifetime of a long-running cluster, but only 256 steps can be ACTIVE or PENDING at any given time. With EMR version 4.8.0 and later, except version 5.0.0, you can cancel steps that are PENDING using the AWS Management Console, the AWS CLI, or the Amazon EMR API.

Adding Steps to a Cluster

You can add steps to a cluster using the AWS CLI, the Amazon EMR SDK, or the AWS Management Console. Using the AWS Management Console, you can add steps to a cluster when the cluster is created. You can also add steps to a long-running cluster—that is, a cluster with the auto-terminate option disabled.

Add Steps Using the Console

Whether you add steps during cluster creation or to a cluster, the procedure is similar to the following.

To add a step to a running cluster using the AWS Management Console

1. In the [Amazon EMR console](#), on the **Cluster List** page, click the link for your cluster.
2. On the **Cluster Details** page, expand the **Steps** section, and then click **Add step**.
3. Type appropriate values in the fields in the **Add Step** dialog, and then click **Add**. Depending on the step type, the options are different.

Add Steps Using the AWS CLI

The following procedures demonstrate adding steps to a newly-created cluster and to a running cluster using the AWS CLI. In both examples, the `--steps` subcommand is used to add steps to the cluster.

To add a step during cluster creation

- Type the following command to create a cluster and add a Pig step. Replace `myKey` with the name of your EC2 key pair and replace `mybucket` with the name of your Amazon S3 bucket.
 - Linux, UNIX, and Mac OS X users:

```
aws emr create-cluster --name "Test cluster" --ami-version 2.4 --applications
Name=Hive Name=Pig \
--use-default-roles --ec2-attributes KeyName=myKey \
--instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m3.xlarge
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m3.xlarge \
--steps Type=PIG,Name="Pig Program",ActionOnFailure=CONTINUE,Args=[-f,s3://mybucket/
scripts/pigscript.pig,-p,INPUT=s3://mybucket/inputdata/,-p,OUTPUT=s3://mybucket/
outputdata/,$INPUT=s3://mybucket/inputdata/,$OUTPUT=s3://mybucket/outputdata/]
```

- Windows users:

```
aws emr create-cluster --name "Test cluster" --ami-version 2.4 --applications
Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --
instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m3.xlarge
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m3.xlarge --steps
Type=PIG,Name="Pig Program",ActionOnFailure=CONTINUE,Args=[-f,s3://mybucket/
scripts/pigscript.pig,-p,INPUT=s3://mybucket/inputdata/,-p,OUTPUT=s3://mybucket/
outputdata/,$INPUT=s3://mybucket/inputdata/,$OUTPUT=s3://mybucket/outputdata/]
```

Note

The list of arguments changes depending on the type of step.

The output is a cluster identifier similar to the following:

```
{
  "ClusterId": "j-2AXXXXXXGAPLF"
}
```

To add a step to a running cluster

- Type the following command to add a step to a running cluster. Replace `j-2AXXXXXXGAPLF` with your cluster ID and replace `mybucket` with your Amazon S3 bucket name.

```
aws emr add-steps --cluster-id j-2AXXXXXXGAPLF --steps Type=PIG,Name="Pig
Program",Args=[-f,s3://mybucket/scripts/pigscript.pig,-p,INPUT=s3://
mybucket/inputdata/,-p,OUTPUT=s3://mybucket/outputdata/,$INPUT=s3://mybucket/
inputdata/,$OUTPUT=s3://mybucket/outputdata/]
```

The output is a step identifier similar to the following:

```
{
  "StepIds": [
    "s-Y9XXXXXXAPMD"
  ]
}
```

For more information on using Amazon EMR commands in the AWS CLI, see <http://docs.aws.amazon.com/cli/latest/reference/emr>.

Viewing Steps

The total number of step records you can view (regardless of status) is 1,000. This total includes both user-submitted and system steps. As the status of user-submitted steps changes to COMPLETED or FAILED, additional user-submitted steps can be added to the cluster until the 1,000 step limit is reached.

After 1,000 steps have been added to a cluster, the submission of additional steps causes the removal of older, user-submitted step records. These records are not removed from the log files. They are removed from the console display, and they do not appear when you use the CLI or API to retrieve cluster information. System step records are never removed.

The step information you can view depends on the mechanism used to retrieve cluster information. The following tables indicate the step information returned by each of the available options.

Option	DescribeJobFlow or --describe --jobflow	ListSteps or list-steps
SDK	256 steps	1,000 steps
Amazon EMR CLI	256 steps	NA
AWS CLI	NA	1,000 steps
API	256 steps	1,000 steps

Cancel Pending Steps

You can cancel steps using the the AWS Management Console, the AWS CLI, or the Amazon EMR API. Only steps that are `PENDING` can be canceled.

To cancel steps using the AWS Management Console

1. In the [Amazon EMR console](#), on the **Cluster List** page, choose the link for the cluster.
2. On the **Cluster Details** page, expand the **Steps** section.
3. For each step you want to cancel, select the step from the list of **Steps**, select **Cancel step**, and then confirm you want to cancel the step.

To cancel steps using the AWS CLI

- Use the `aws emr cancel-steps` command, specifying the cluster and steps to cancel. The following example demonstrates an AWS CLI command to cancel two steps.

```
aws emr cancel-steps --ClusterID j-2QUAJ7T3OTEI8 --
StepIDs s-3M8DKCZYYN1QE,s-3M8DKCZYYN1QE
```

Submit Hadoop Jobs Interactively

In addition to adding steps to a cluster, you can connect to the master node using an SSH client or the AWS CLI and interactively submit Hadoop jobs. For example, you can use PuTTY to establish an SSH connection with the master node and submit interactive Hive queries which are compiled into one or more Hadoop jobs.

You can submit Hadoop jobs interactively by establishing an SSH connection to the master node (using an SSH client such as PuTTY or OpenSSH) or by using the `ssh` subcommand in the AWS CLI. You can submit jobs interactively to the master node even if you have 256 active steps running on the cluster. Note however that log records associated with interactively submitted jobs are included in the "step created jobs" section of the currently running step's controller log. For more information about step logs, see [View Log Files](#) (p. 146).

The following examples demonstrate interactively submitting Hadoop jobs and Hive jobs to the master node. The process for submitting jobs for other programming frameworks (such as Pig) is similar to these examples.

To submit Hadoop jobs interactively using the AWS CLI

- You can submit Hadoop jobs interactively using the AWS CLI by establishing an SSH connection in the CLI command (using the `ssh` subcommand). To copy a JAR file from your local Windows machine to the master node's file system, type the following command. Replace `j-2A6HXXXXXXL7J` with your cluster ID, replace `mykey.ppk` with the name of your key pair file, and replace `myjar.jar` with the name of your JAR file.

```
aws emr put --cluster-id j-2A6HXXXXXXL7J --key-pair-file "C:\Users\username\Desktop\Keys\mykey.ppk" --src "C:\Users\username\myjar.jar"
```

To create an SSH connection and submit the Hadoop job `myjar.jar`, type the following command.

```
aws emr ssh --cluster-id j-2A6HXXXXXXL7J --key-pair-file "C:\Users\username\Desktop\Keys\mykey.ppk" --command "hadoop jar myjar.jar"
```

For more information on using Amazon EMR commands in the AWS CLI, see <http://docs.aws.amazon.com/cli/latest/reference/emr>.

To interactively submit Hive jobs using the AWS CLI

In addition to submitting jobs to the master node via JAR files, you can submit jobs by interacting with one of the Hadoop programming frameworks running on the master node. For example, you can interactively submit Hive queries or Pig transformations at the command line, or you can submit scripts to the cluster for processing. Your commands or scripts are then compiled into one or more Hadoop jobs.

The following procedure demonstrates running a Hive script using the AWS CLI.

1. If Hive is not installed on the cluster, type the following command to install it. Replace `j-2A6HXXXXXXL7J` with your cluster ID.

```
aws emr install-applications --cluster-id j-2A6HXXXXXXL7J --apps Name=Hive
```

2. Create a Hive script file containing the queries or commands to run. The following example script named `my-hive.q` creates two tables, `aTable` and `anotherTable`, and copies the contents of `aTable` to `anotherTable`, replacing all data.

```
---- sample Hive script file: my-hive.q ----
create table aTable (aColumn string) ;
create table anotherTable like aTable;
insert overwrite table anotherTable select * from aTable
```

3. Type the following commands to run the script from the command line using the `ssh` subcommand.

To copy `my-hive.q` from a Windows machine to your cluster, type the following command. Replace `j-2A6HXXXXXXL7J` with your cluster ID and replace `mykey.ppk` with the name of your key pair file.

```
aws emr put --cluster-id j-2A6HXXXXXXL7J --key-pair-file "C:\Users\username\Desktop\Keys\mykey.ppk" --src "C:\Users\username\my-hive.q"
```

To create an SSH connection and submit the Hive script `my-hive.q`, type the following command.

```
aws emr ssh --cluster-id j-2A6HXXXXXL7J --key-pair-file "C:\Users\username\Desktop\Keys\mykey.ppk" --command "hive -f my-hive.q"
```

For more information on using Amazon EMR commands in the AWS CLI, see <http://docs.aws.amazon.com/cli/latest/reference/emr>.

Add More than 256 Steps to a Cluster

Beginning with AMI 3.1.1 (Hadoop 2.x) and AMI 2.4.8 (Hadoop 1.x), you can submit an unlimited number of steps over the lifetime of a long-running cluster, but only 256 can be active or pending at any given time. For earlier AMI versions, the total number of steps that can be processed by a cluster is limited to 256 (including system steps such as install Hive and install Pig). For more information, see [Submit Work to a Cluster](#) (p. 206).

You can use one of several methods to overcome the 256 step limit in pre-3.1.1 and pre-2.4.8 AMIs:

1. Have each step submit several jobs to Hadoop. This does not allow you unlimited steps in pre-3.1.1 and pre-2.4.8 AMIs, but it is the easiest solution if you need a fixed number of steps greater than 256.
2. Write a workflow program that runs in a step on a long-running cluster and submits jobs to Hadoop. You could have the workflow program either:
 - Listen to an Amazon SQS queue to receive information about new steps to run.
 - Check an Amazon S3 bucket on a regular schedule for files containing information about the new steps to run.
3. Write a workflow program that runs on an EC2 instance outside of Amazon EMR and submits jobs to your long-running clusters using SSH.
4. Connect to your long-running cluster via SSH and submit Hadoop jobs using the Hadoop API. For more information, see <http://hadoop.apache.org/docs/current/api/org/apache/hadoop/mapred/JobClient.html>.
5. Connect to the master node using an SSH client (such as PuTTY or OpenSSH) and manually submit jobs to the cluster or use the `ssh` subcommand in the AWS CLI to both connect and submit jobs. For more information about establishing an SSH connection with the master node, see [Connect to the Master Node Using SSH](#) (p. 173). For more information about interactively submitting Hadoop jobs, see [Submit Hadoop Jobs Interactively](#) (p. 208).

Automate Recurring Clusters with AWS Data Pipeline

AWS Data Pipeline is a service that automates the movement and transformation of data. You can use it to schedule moving input data into Amazon S3 and to schedule launching clusters to process that data. For example, consider the case where you have a web server recording traffic logs. If you want to run a weekly cluster to analyze the traffic data, you can use AWS Data Pipeline to schedule those clusters. AWS Data Pipeline is a data-driven workflow, so that one task (launching the cluster) can be dependent on another task (moving the input data to Amazon S3). It also has robust retry functionality.

For more information about AWS Data Pipeline, see the [AWS Data Pipeline Developer Guide](#), especially the tutorials regarding Amazon EMR:

- [Tutorial: Launch an Amazon EMR Job Flow](#)
- [Getting Started: Process Web Logs with AWS Data Pipeline, Amazon EMR, and Hive](#)
- [Tutorial: Amazon DynamoDB Import and Export Using AWS Data Pipeline](#)

Troubleshoot a Cluster

This documentation is for versions 4.x and 5.x of Amazon EMR. For information about Amazon EMR AMI versions 2.x and 3.x, see the [Amazon EMR Developer Guide \(PDF\)](#).

A cluster hosted by Amazon EMR runs in a complex ecosystem made up of several types of open-source software, custom application code, and Amazon Web Services. An issue in any of these parts can cause the cluster to fail or take longer than expected to complete. The following topics will help you figure out what has gone wrong in your cluster and give you suggestions on how to fix it.

Topics

- [What Tools are Available for Troubleshooting? \(p. 211\)](#)
- [Troubleshoot a Failed Cluster \(p. 212\)](#)
- [Troubleshoot a Slow Cluster \(p. 216\)](#)
- [Common Errors in Amazon EMR \(p. 222\)](#)

When you are developing a new Hadoop application, we recommend that you enable debugging and process a small but representative subset of your data to test the application. You may also want to run the application step-by-step to test each step separately. For more information, see [Configure Cluster Logging and Debugging \(p. 120\)](#) and [Step 5: Test the Cluster Step by Step \(p. 215\)](#).

What Tools are Available for Troubleshooting?

There are several tools you can use to gather information about your cluster to help determine what went wrong. Some require that you initialize them when you launch the cluster; others are available for every cluster.

Topics

- [Tools to Display Cluster Details \(p. 211\)](#)
- [Tools to View Log Files \(p. 212\)](#)
- [Tools to Monitor Cluster Performance \(p. 212\)](#)

Tools to Display Cluster Details

You can use any of the Amazon EMR interfaces (console, CLI or API) to retrieve detailed information about a cluster. For more information, see [View Cluster Details \(p. 140\)](#).

Amazon EMR Console Details Pane

The console displays all of the clusters you've launched in the past two weeks, regardless of whether they are active or terminated. If you click on a cluster, the console displays a details pane with information about that cluster.

Amazon EMR Command Line Interface

You can locate details about a cluster from the CLI using the `--describe` argument.

Amazon EMR API

You can locate details about a cluster from the API using the `DescribeJobFlows` action.

Tools to View Log Files

Amazon EMR and Hadoop both generate log files as the cluster runs. You can access these log files from several different tools, depending on the configuration you specified when you launched the cluster. For more information, see [Configure Cluster Logging and Debugging \(p. 120\)](#).

Log Files on the Master Node

Every cluster publishes logs files to the `/mnt/var/log/` directory on the master node. These log files are only available while the cluster is running.

Log Files Archived to Amazon S3

If you launch the cluster and specify an Amazon S3 log path, the cluster copies the log files stored in `/mnt/var/log/` on the master node to Amazon S3 in 5-minute intervals. This ensures that you have access to the log files even after the cluster is terminated. Because the files are archived in 5-minute intervals, the last few minutes of an suddenly terminated cluster may not be available.

Tools to Monitor Cluster Performance

Amazon EMR provides several tools to monitor the performance of your cluster.

Hadoop Web Interfaces

Every cluster publishes a set of web interfaces on the master node that contain information about the cluster. You can access these web pages by using an SSH tunnel to connect them on the master node. For more information, see [View Web Interfaces Hosted on Amazon EMR Clusters \(p. 177\)](#).

CloudWatch Metrics

Every cluster reports metrics to CloudWatch. CloudWatch is a web service that tracks metrics, and which you can use to set alarms on those metrics. For more information, see [Monitor Metrics with CloudWatch \(p. 157\)](#).

Troubleshoot a Failed Cluster

This section walks you through the process of troubleshooting a cluster that has failed. This means that the cluster terminated with an error code. If the cluster is still running, but is taking a long time to return results, see [Troubleshoot a Slow Cluster \(p. 216\)](#) instead.

Topics

- [Step 1: Gather Data About the Issue \(p. 213\)](#)
- [Step 2: Check the Environment \(p. 213\)](#)
- [Step 3: Look at the Last State Change \(p. 214\)](#)
- [Step 4: Examine the Log Files \(p. 214\)](#)
- [Step 5: Test the Cluster Step by Step \(p. 215\)](#)

Step 1: Gather Data About the Issue

The first step in troubleshooting an cluster is to gather information about what went wrong and the current status and configuration of the cluster. This information will be used in the following steps to confirm or rule out possible causes of the issue.

Define the Problem

A clear definition of the problem is the first place to begin. Some questions to ask yourself:

- What did I expect to happen? What happened instead?
- When did this problem first occur? How often has it happened since?
- Has anything changed in how I configure or run my cluster?

Cluster Details

The following cluster details are useful in helping track down issues. For more information on how to gather this information, see [View Cluster Details \(p. 140\)](#).

- Identifier of the cluster. (Also called a job flow identifier.)
- Region and availability zone the cluster was launched into.
- State of the cluster, including details of the last state change.
- Type and number of EC2 instances specified for the master, core, and task nodes.

Step 2: Check the Environment

Amazon EMR operates as part of an ecosystem of web services and open-source software. Things that affect those dependencies can impact the performance of Amazon EMR.

Topics

- [Check for Service Outages \(p. 213\)](#)
- [Check Usage Limits \(p. 214\)](#)
- [Check the Release Version \(p. 214\)](#)
- [Check the Amazon VPC Subnet Configuration \(p. 214\)](#)

Check for Service Outages

Amazon EMR uses several Amazon Web Services internally. It runs virtual servers on Amazon EC2, stores data and scripts on Amazon S3, indexes log files in Amazon SimpleDB, and reports metrics to CloudWatch. Events that disrupt these services are rare — but when they occur — can cause issues in Amazon EMR.

Before you go further, check the [Service Health Dashboard](#). Check the region where you launched your cluster to see whether there are disruption events in any of these services.

Check Usage Limits

If you are launching a large cluster, have launched many clusters simultaneously, or you are an IAM user sharing an AWS account with other users, the cluster may have failed because you exceeded an AWS service limit.

Amazon EC2 limits the number of virtual server instances running on a single AWS region to 20 on-demand or reserved instances. If you launch a cluster with more than 20 nodes, or launch a cluster that causes the total number of EC2 instances active on your AWS account to exceed 20, the cluster will not be able to launch all of the EC2 instances it requires and may fail. When this happens, Amazon EMR returns an `EC2_QUOTA_EXCEEDED` error. You can request that AWS increase the number of EC2 instances that you can run on your account by submitting a [Request to Increase Amazon EC2 Instance Limit](#) application.

Another thing that may cause you to exceed your usage limits is the delay between when a cluster is terminated and when it releases all of its resources. Depending on its configuration, it may take up to 5-20 minutes for a cluster to fully terminate and release allocated resources. If you are getting an `EC2_QUOTA_EXCEEDED` error when you attempt to launch a cluster, it may be because resources from a recently terminated cluster may not yet have been released. In this case, you can either [request that your Amazon EC2 quota be increased](#), or you can wait twenty minutes and re-launch the cluster.

Amazon S3 limits the number of buckets created on an account to 100. If your cluster creates a new bucket that exceeds this limit, the bucket creation will fail and may cause the cluster to fail.

Check the Release Version

Compare the release label that you used to launch the cluster with the latest Amazon EMR release. Each release of Amazon EMR includes improvements such as new applications, features, patches, and bug fixes. The issue that is affecting your cluster may have already been fixed in the latest release version. If possible, re-run your cluster using the latest version.

Check the Amazon VPC Subnet Configuration

If your cluster was launched in a Amazon VPC subnet, the subnet needs to be configured as described in [Plan and Configure Networking \(p. 73\)](#). In addition, check that the subnet you launch the cluster into has enough free elastic IP addresses to assign one to each node in the cluster.

Step 3: Look at the Last State Change

The last state change provides information about what occurred the last time the cluster changed state. This often has information that can tell you what went wrong as a cluster changes state to `FAILED`. For example, if you launch a streaming cluster and specify an output location that already exists in Amazon S3, the cluster will fail with a last state change of "Streaming output directory already exists".

You can locate the last state change value from the console by viewing the details pane for the cluster, from the CLI using the `list-steps` or `describe-cluster` arguments, or from the API using the `DescribeCluster` and `ListSteps` actions. For more information, see [View Cluster Details \(p. 140\)](#).

Step 4: Examine the Log Files

The next step is to examine the log files in order to locate an error code or other indication of the issue that your cluster experienced. For information on the log files available, where to find them, and how to view them, see [View Log Files \(p. 146\)](#).

It may take some investigative work to determine what happened. Hadoop runs the work of the jobs in task attempts on various nodes in the cluster. Amazon EMR can initiate speculative task attempts, terminating the other task attempts that do not complete first. This generates significant activity that is logged to the controller, stderr and syslog log files as it happens. In addition, multiple tasks attempts are running simultaneously, but a log file can only display results linearly.

Start by checking the bootstrap action logs for errors or unexpected configuration changes during the launch of the cluster. From there, look in the step logs to identify Hadoop jobs launched as part of a step with errors. Examine the Hadoop job logs to identify the failed task attempts. The task attempt log will contain details about what caused a task attempt to fail.

The following sections describe how to use the various log files to identify error in your cluster.

Check the Bootstrap Action Logs

Bootstrap actions run scripts on the cluster as it is launched. They are commonly used to install additional software on the cluster or to alter configuration settings from the default values. Checking these logs may provide insight into errors that occurred during set up of the cluster as well as configuration settings changes that could affect performance.

Check the Step Logs

There are four types of step logs.

- **controller**—Contains files generated by Amazon EMR (Amazon EMR) that arise from errors encountered while trying to run your step. If your step fails while loading, you can find the stack trace in this log. Errors loading or accessing your application are often described here, as are missing mapper file errors.
- **stderr**—Contains error messages that occurred while processing the step. Application loading errors are often described here. This log sometimes contains a stack trace.
- **stdout**—Contains status generated by your mapper and reducer executables. Application loading errors are often described here. This log sometimes contains application error messages.
- **syslog**—Contains logs from non-Apache software, such as Apache and Hadoop. Streaming errors are often described here.

Check stderr for obvious errors. If stderr displays a short list of errors, the step came to a quick stop with an error thrown. This is most often caused by an error in the mapper and reducer applications being run in the cluster.

Examine the last lines of controller and syslog for notices of errors or failures. Follow any notices about failed tasks, particularly if it says "Job Failed".

Check the Task Attempt Logs

If the previous analysis of the step logs turned up one or more failed tasks, investigate the logs of the corresponding task attempts for more detailed error information.

Step 5: Test the Cluster Step by Step

A useful technique when you are trying to track down the source of an error is to restart the cluster and submit the steps to it one by one. This lets you check the results of each step before processing the next one, and gives you the opportunity to correct and re-run a step that has failed. This also has the advantage that you only load your input data once.

To test a cluster step by step

1. Launch a new cluster, with both keep alive and termination protection enabled. Keep alive keeps the cluster running after it has processed all of its pending steps. Termination protection prevents a cluster from shutting down in the event of an error. For more information, see [Configure a Cluster to be Transient or Long-Running \(p. 56\)](#) and [Managing Cluster Termination \(p. 188\)](#).
2. Submit a step to the cluster. For more information, see [Submit Work to a Cluster \(p. 206\)](#).
3. When the step completes processing, check for errors in the step log files. For more information, see [Step 4: Examine the Log Files \(p. 214\)](#). The fastest way to locate these log files is by connecting to the master node and viewing the log files there. The step log files do not appear until the step runs for some time, finishes, or fails.
4. If the step succeeded without error, run the next step. If there were errors, investigate the error in the log files. If it was an error in your code, make the correction and re-run the step. Continue until all steps run without error.
5. When you are done debugging the cluster, and want to terminate it, you will have to manually terminate it. This is necessary because the cluster was launched with termination protection enabled. For more information, see [Managing Cluster Termination \(p. 188\)](#).

Troubleshoot a Slow Cluster

This section walks you through the process of troubleshooting a cluster that is still running, but is taking a long time to return results. For more information about what to do if the cluster has terminated with an error code, see [Troubleshoot a Failed Cluster \(p. 212\)](#)

Amazon EMR enables you to specify the number and kind of instances in the cluster. These specifications are the primary means of affecting the speed with which your data processing completes. One thing you might consider is re-running the cluster, this time specifying EC2 instances with greater resources, or specifying a larger number of instances in the cluster. For more information, see [Configure Cluster Hardware and Networking \(p. 67\)](#).

The following topics walk you through the process of identifying alternative causes of a slow cluster.

Topics

- [Step 1: Gather Data About the Issue \(p. 216\)](#)
- [Step 2: Check the Environment \(p. 217\)](#)
- [Step 3: Examine the Log Files \(p. 218\)](#)
- [Step 4: Check Cluster and Instance Health \(p. 219\)](#)
- [Step 5: Check for Arrested Groups \(p. 220\)](#)
- [Step 6: Review Configuration Settings \(p. 220\)](#)
- [Step 7: Examine Input Data \(p. 222\)](#)

Step 1: Gather Data About the Issue

The first step in troubleshooting an cluster is to gather information about what went wrong and the current status and configuration of the cluster. This information will be used in the following steps to confirm or rule out possible causes of the issue.

Define the Problem

A clear definition of the problem is the first place to begin. Some questions to ask yourself:

- What did I expect to happen? What happened instead?
- When did this problem first occur? How often has it happened since?
- Has anything changed in how I configure or run my cluster?

Cluster Details

The following cluster details are useful in helping track down issues. For more information on how to gather this information, see [View Cluster Details \(p. 140\)](#).

- Identifier of the cluster. (Also called a job flow identifier.)
- Region and availability zone the cluster was launched into.
- State of the cluster, including details of the last state change.
- Type and number of EC2 instances specified for the master, core, and task nodes.

Step 2: Check the Environment

Topics

- [Check for Service Outages \(p. 217\)](#)
- [Check Usage Limits \(p. 217\)](#)
- [Check the Amazon VPC Subnet Configuration \(p. 218\)](#)
- [Restart the Cluster \(p. 218\)](#)

Check for Service Outages

Amazon EMR uses several Amazon Web Services internally. It runs virtual servers on Amazon EC2, stores data and scripts on Amazon S3, indexes log files in Amazon SimpleDB, and reports metrics to CloudWatch. Events that disrupt these services are rare — but when they occur — can cause issues in Amazon EMR.

Before you go further, check the [Service Health Dashboard](#). Check the region where you launched your cluster to see whether there are disruption events in any of these services.

Check Usage Limits

If you are launching a large cluster, have launched many clusters simultaneously, or you are an IAM user sharing an AWS account with other users, the cluster may have failed because you exceeded an AWS service limit.

Amazon EC2 limits the number of virtual server instances running on a single AWS region to 20 on-demand or reserved instances. If you launch a cluster with more than 20 nodes, or launch a cluster that causes the total number of EC2 instances active on your AWS account to exceed 20, the cluster will not be able to launch all of the EC2 instances it requires and may fail. When this happens, Amazon EMR returns an `EC2_QUOTA_EXCEEDED` error. You can request that AWS increase the number of EC2 instances that you can run on your account by submitting a [Request to Increase Amazon EC2 Instance Limit](#) application.

Another thing that may cause you to exceed your usage limits is the delay between when a cluster is terminated and when it releases all of its resources. Depending on its configuration, it may take up to 5-20 minutes for a cluster to fully terminate and release allocated resources. If you are getting an `EC2_QUOTA_EXCEEDED` error when you attempt to launch a cluster, it may be because resources from a recently

terminated cluster may not yet have been released. In this case, you can either [request that your Amazon EC2 quota be increased](#), or you can wait twenty minutes and re-launch the cluster.

Amazon S3 limits the number of buckets created on an account to 100. If your cluster creates a new bucket that exceeds this limit, the bucket creation will fail and may cause the cluster to fail.

Check the Amazon VPC Subnet Configuration

If your cluster was launched in a Amazon VPC subnet, the subnet needs to be configured as described in [Plan and Configure Networking \(p. 73\)](#). In addition, check that the subnet you launch the cluster into has enough free elastic IP addresses to assign one to each node in the cluster.

Restart the Cluster

The slow down in processing may be caused by a transient condition. Consider terminating and restarting the cluster to see if performance improves.

Step 3: Examine the Log Files

The next step is to examine the log files in order to locate an error code or other indication of the issue that your cluster experienced. For information on the log files available, where to find them, and how to view them, see [View Log Files \(p. 146\)](#).

It may take some investigative work to determine what happened. Hadoop runs the work of the jobs in task attempts on various nodes in the cluster. Amazon EMR can initiate speculative task attempts, terminating the other task attempts that do not complete first. This generates significant activity that is logged to the controller, stderr and syslog log files as it happens. In addition, multiple tasks attempts are running simultaneously, but a log file can only display results linearly.

Start by checking the bootstrap action logs for errors or unexpected configuration changes during the launch of the cluster. From there, look in the step logs to identify Hadoop jobs launched as part of a step with errors. Examine the Hadoop job logs to identify the failed task attempts. The task attempt log will contain details about what caused a task attempt to fail.

The following sections describe how to use the various log files to identify error in your cluster.

Check the Bootstrap Action Logs

Bootstrap actions run scripts on the cluster as it is launched. They are commonly used to install additional software on the cluster or to alter configuration settings from the default values. Checking these logs may provide insight into errors that occurred during set up of the cluster as well as configuration settings changes that could affect performance.

Check the Step Logs

There are four types of step logs.

- **controller**—Contains files generated by Amazon EMR (Amazon EMR) that arise from errors encountered while trying to run your step. If your step fails while loading, you can find the stack trace in this log. Errors loading or accessing your application are often described here, as are missing mapper file errors.
- **stderr**—Contains error messages that occurred while processing the step. Application loading errors are often described here. This log sometimes contains a stack trace.
- **stdout**—Contains status generated by your mapper and reducer executables. Application loading errors are often described here. This log sometimes contains application error messages.

- **syslog**—Contains logs from non-Amazon software, such as Apache and Hadoop. Streaming errors are often described here.

Check stderr for obvious errors. If stderr displays a short list of errors, the step came to a quick stop with an error thrown. This is most often caused by an error in the mapper and reducer applications being run in the cluster.

Examine the last lines of controller and syslog for notices of errors or failures. Follow any notices about failed tasks, particularly if it says "Job Failed".

Check the Task Attempt Logs

If the previous analysis of the step logs turned up one or more failed tasks, investigate the logs of the corresponding task attempts for more detailed error information.

Check the Hadoop Daemon Logs

In rare cases, Hadoop itself might fail. To see if that is the case, you must look at the Hadoop logs. They are located at `/var/log/hadoop/` on each node.

You can use the JobTracker logs to map a failed task attempt to the node it was run on. Once you know the node associated with the task attempt, you can check the health of the EC2 instance hosting that node to see if there were any issues such as running out of CPU or memory.

Step 4: Check Cluster and Instance Health

An Amazon EMR cluster is made up of nodes running on Amazon EC2 instances. If those instances become resource-bound (such as running out of CPU or memory), experience network connectivity issues, or are terminated, the speed of cluster processing suffers.

There are up to three types of nodes in a cluster:

- **master node** — manages the cluster. If it experiences a performance issue, the entire cluster is affected.
- **core nodes** — process map-reduce tasks and maintain the Hadoop Distributed Filesystem (HDFS). If one of these nodes experiences a performance issue, it can slow down HDFS operations as well as map-reduce processing. You can add additional core nodes to a cluster to improve performance, but cannot remove core nodes. For more information, see [Manually Resizing a Running Cluster \(p. 199\)](#).
- **task nodes** — process map-reduce tasks. These are purely computational resources and do not store data. You can add task nodes to a cluster to speed up performance, or remove task nodes that are not needed. For more information, see [Manually Resizing a Running Cluster \(p. 199\)](#).

When you look at the health of a cluster, you should look at both the performance of the cluster overall, as well as the performance of individual instances. There are several tools you can use:

Check Cluster Health with CloudWatch

Every Amazon EMR cluster reports metrics to CloudWatch. These metrics provide summary performance information about the cluster, such as the total load, HDFS utilization, running tasks, remaining tasks, corrupt blocks, and more. Looking at the CloudWatch metrics gives you the big picture about what is going on with your cluster and can provide insight into what is causing the slow down in processing. In addition to using CloudWatch to analyze an existing performance issue, you can set alarms that cause CloudWatch to alert if a future performance issue occurs. For more information, see [Monitor Metrics with CloudWatch \(p. 157\)](#).

Check Job and HDFS Health with Hadoop Web Interfaces

Hadoop provides a series of web interfaces you can use to view information. For more information about how to access these web interfaces, see [View Web Interfaces Hosted on Amazon EMR Clusters \(p. 177\)](#).

- JobTracker — provides information about the progress of job being processed by the cluster. You can use this interface to identify when a job has become stuck.
- HDFS NameNode — provides information about the percentage of HDFS utilization and available space on each node. You can use this interface to identify when HDFS is becoming resource bound and requires additional capacity.
- TaskTracker — provides information about the tasks of the job being processed by the cluster. You can use this interface to identify when a task has become stuck.

Check Instance Health with Amazon EC2

Another way to look for information about the status of the instances in your cluster is to use the Amazon EC2 console. Because each node in the cluster runs on an EC2 instance, you can use tools provided by Amazon EC2 to check their status. For more information, see [View Cluster Instances in Amazon EC2 \(p. 150\)](#).

Step 5: Check for Arrested Groups

An instance group becomes arrested when it encounters too many errors while trying to launch nodes. For example, if new nodes repeatedly fail while performing bootstrap actions, the instance group will — after some time — go into the `ARRESTED` state rather than continuously attempt to provision new nodes.

A node could fail to come up if:

- Hadoop or the cluster is somehow broken and does not accept a new node into the cluster
- A bootstrap action fails on the new node
- The node is not functioning correctly and fails to check in with Hadoop

If an instance group is in the `ARRESTED` state, and the cluster is in a `WAITING` state, you can add a cluster step to reset the desired number of slave nodes. Adding the step resumes processing of the cluster and put the instance group back into a `RUNNING` state.

For more information about how to reset a cluster in an arrested state, see [Arrested State \(p. 202\)](#).

Step 6: Review Configuration Settings

Configuration settings specify details about how a cluster runs, such as how many times to retry a task and how much memory is available for sorting. When you launch a cluster using Amazon EMR, there are Amazon EMR-specific settings in addition to the standard Hadoop configuration settings. The configuration settings are stored on the master node of the cluster. You can check the configuration settings to ensure that your cluster has the resources it requires to run efficiently.

Amazon EMR defines default Hadoop configuration settings that it uses to launch a cluster. The values are based on the AMI and the instance type you specify for the cluster. You can modify the configuration settings from the default values using a bootstrap action or by specifying new values in job execution parameters. For more information, see [\(Optional\) Create Bootstrap Actions to Install Additional Software \(p. 64\)](#). To determine whether a bootstrap action changed the configuration settings, check the bootstrap action logs.

Amazon EMR logs the Hadoop settings used to execute each job. The log data is stored in a file named `job_<job-id>_conf.xml` under the `/mnt/var/log/hadoop/history/` directory of the master node, where

`job-id` is replaced by the identifier of the job. If you've enabled log archiving, this data is copied to Amazon S3 in the `logs/date/jobflow-id/jobs` folder, where `date` is the date the job ran, and `jobflow-id` is the identifier of the cluster.

The following Hadoop job configuration settings are especially useful for investigating performance issues. For more information about the Hadoop configuration settings and how they affect the behavior of Hadoop, go to <http://hadoop.apache.org/docs/>.

Configuration Setting	Description
<code>dfs.replication</code>	The number of HDFS nodes to which a single block (like the hard drive block) is copied to in order to produce a RAID-like environment. Determines the number of HDFS nodes which contain a copy of the block.
<code>io.sort.mb</code>	Total memory available for sorting. This value should be 10x <code>io.sort.factor</code> . This setting can also be used to calculate total memory used by task node by figuring <code>io.sort.mb</code> multiplied by <code>mapred.tasktracker.ap.tasks.maximum</code> .
<code>io.sort.spill.percent</code>	Used during sort, at which point the disk will start to be used because the allotted sort memory is getting full.
<code>mapred.child.java.opts</code>	Deprecated. Use <code>mapred.map.child.java.opts</code> and <code>mapred.reduce.child.java.opts</code> instead. The Java options TaskTracker uses when launching a JVM for a task to execute within. A common parameter is <code>"-Xmx"</code> for setting max memory size.
<code>mapred.map.child.java.opts</code>	The Java options TaskTracker uses when launching a JVM for a map task to execute within. A common parameter is <code>"-Xmx"</code> for setting max memory heap size.
<code>mapred.map.tasks.speculative.execution</code>	Determines whether map task attempts of the same task may be launched in parallel.
<code>mapred.reduce.tasks.speculative.execution</code>	Determines whether reduce task attempts of the same task may be launched in parallel.
<code>mapred.map.max.attempts</code>	The maximum number of times a map task can be attempted. If all fail, then the map task is marked as failed.
<code>mapred.reduce.child.java.opts</code>	The Java options TaskTracker uses when launching a JVM for a reduce task to execute within. A common parameter is <code>"-Xmx"</code> for setting max memory heap size.
<code>mapred.reduce.max.attempts</code>	The maximum number of times a reduce task can be attempted. If all fail, then the map task is marked as failed.
<code>mapred.reduce.slowstart.completed.maps</code>	The amount of maps tasks that should complete before reduce tasks are attempted. Not waiting long enough may cause "Too many fetch-failure" errors in attempts.
<code>mapred.reuse.jvm.num.tasks</code>	A task runs within a single JVM. Specifies how many tasks may reuse the same JVM.
<code>mapred.tasktracker.map.tasks.maximum</code>	The max amount of tasks that can execute in parallel per task node during mapping.

Configuration Setting	Description
mapred.tasktracker.reduce.tasks.maximum	The max amount of tasks that can execute in parallel per task node during reducing.

If your cluster tasks are memory-intensive, you can enhance performance by using fewer tasks per core node and reducing your job tracker heap size.

Step 7: Examine Input Data

Look at your input data. Is it distributed evenly among your key values? If your data is heavily skewed towards one or few key values, the processing load may be mapped to a small number of nodes, while other nodes idle. This imbalanced distribution of work can result in slower processing times.

An example of an imbalanced data set would be running a cluster to alphabetize words, but having a data set that contained only words beginning with the letter "a". When the work was mapped out, the node processing values beginning with "a" would be overwhelmed, while nodes processing words beginning with other letters would go idle.

Common Errors in Amazon EMR

There are many reasons why a cluster might fail or be slow in processing data. The following sections list the most common issues and suggestions for fixing them.

Topics

- [Input and Output Errors \(p. 222\)](#)
- [Permissions Errors \(p. 224\)](#)
- [Resource Errors \(p. 225\)](#)
- [Streaming Cluster Errors \(p. 228\)](#)
- [Custom JAR Cluster Errors \(p. 229\)](#)
- [Hive Cluster Errors \(p. 230\)](#)
- [VPC Errors \(p. 231\)](#)
- [AWS GovCloud \(US\) Errors \(p. 233\)](#)
- [Other Issues \(p. 234\)](#)

Input and Output Errors

The following errors are common in cluster input and output operations.

Topics

- [Does your path to Amazon Simple Storage Service \(Amazon S3\) have at least three slashes? \(p. 223\)](#)
- [Are you trying to recursively traverse input directories? \(p. 223\)](#)
- [Does your output directory already exist? \(p. 223\)](#)
- [Are you trying to specify a resource using an HTTP URL? \(p. 223\)](#)
- [Are you referencing an Amazon S3 bucket using an invalid name format? \(p. 223\)](#)
- [Are you experiencing trouble loading data to or from Amazon S3? \(p. 223\)](#)

Does your path to Amazon Simple Storage Service (Amazon S3) have at least three slashes?

When you specify an Amazon S3 bucket, you must include a terminating slash on the end of the URL. For example, instead of referencing a bucket as `"s3n://myawsbucket"`, you should use `"s3n://myawsbucket/"`, otherwise Hadoop fails your cluster in most cases.

Are you trying to recursively traverse input directories?

Hadoop does not recursively search input directories for files. If you have a directory structure such as `/corpus/01/01.txt`, `/corpus/01/02.txt`, `/corpus/02/01.txt`, etc. and you specify `/corpus/` as the input parameter to your cluster, Hadoop does not find any input files because the `/corpus/` directory is empty and Hadoop does not check the contents of the subdirectories. Similarly, Hadoop does not recursively check the subdirectories of Amazon S3 buckets.

The input files must be directly in the input directory or Amazon S3 bucket that you specify, not in subdirectories.

Does your output directory already exist?

If you specify an output path that already exists, Hadoop will fail the cluster in most cases. This means that if you run a cluster one time and then run it again with exactly the same parameters, it will likely work the first time and then never again; after the first run, the output path exists and thus causes all successive runs to fail.

Are you trying to specify a resource using an HTTP URL?

Hadoop does not accept resource locations specified using the `http://` prefix. You cannot reference a resource using an HTTP URL. For example, passing in `http://mysite/myjar.jar` as the JAR parameter causes the cluster to fail.

Are you referencing an Amazon S3 bucket using an invalid name format?

If you attempt to use a bucket name such as `"myawsbucket.1"` with Amazon EMR, your cluster will fail because Amazon EMR requires that bucket names be valid RFC 2396 host names; the name cannot end with a number. In addition, because of the requirements of Hadoop, Amazon S3 bucket names used with Amazon EMR must contain only lowercase letters, numbers, periods (`.`), and hyphens (`-`). For more information about how to format Amazon S3 bucket names, see [Bucket Restrictions and Limitations](#) in the *Amazon Simple Storage Service Developer Guide*.

Are you experiencing trouble loading data to or from Amazon S3?

Amazon S3 is the most popular input and output source for Amazon EMR. A common mistake is to treat Amazon S3 as you would a typical file system. There are differences between Amazon S3 and a file system that you need to take into account when running your cluster.

- If an internal error occurs in Amazon S3, your application needs to handle this gracefully and re-try the operation.
- If calls to Amazon S3 take too long to return, your application may need to reduce the frequency at which it calls Amazon S3.
- Listing all the objects in an Amazon S3 bucket is an expensive call. Your application should minimize the number of times it does this.

There are several ways you can improve how your cluster interacts with Amazon S3.

- Use S3DistCp to move objects in and out of Amazon S3. S3DistCp implements error handling, retries and back-offs to match the requirements of Amazon S3. For more information, see [Distributed Copy Using S3DistCp](#).
- Design your application with eventual consistency in mind. Use HDFS for intermediate data storage while the cluster is running and Amazon S3 only to input the initial data and output the final results.
- If your clusters will commit 200 or more transactions per second to Amazon S3, [contact support](#) to prepare your bucket for greater transactions per second and consider using the key partition strategies described in [Amazon S3 Performance Tips & Tricks](#).
- Set the Hadoop configuration setting `io.file.buffer.size` to 65536. This causes Hadoop to spend less time seeking through Amazon S3 objects.
- Consider disabling Hadoop's speculative execution feature if your cluster is experiencing Amazon S3 concurrency issues. You do this through the `mapred.map.tasks.speculative.execution` and `mapred.reduce.tasks.speculative.execution` configuration settings. This is also useful when you are troubleshooting a slow cluster.
- If you are running a Hive cluster, see [Are you having trouble loading data to or from Amazon S3 into Hive? \(p. 230\)](#).

For additional information, see [Amazon S3 Error Best Practices](#) in the *Amazon Simple Storage Service Developer Guide*.

Permissions Errors

The following errors are common when using permissions or credentials.

Topics

- [Are you passing the correct credentials into SSH? \(p. 224\)](#)
- [If you are using IAM, do you have the proper Amazon EC2 policies set? \(p. 225\)](#)

Are you passing the correct credentials into SSH?

If you are unable to use SSH to connect to the master node, it is most likely an issue with your security credentials.

First, check that the `.pem` file containing your SSH key has the proper permissions. You can use `chmod` to change the permissions on your `.pem` file as is shown in the following example, where you would replace `mykey.pem` with the name of your own `.pem` file.

```
chmod og-rwx mykey.pem
```

The second possibility is that you are not using the keypair you specified when you created the cluster. This is easy to do if you have created multiple key pairs. Check the cluster details in the Amazon EMR console (or use the `--describe` option in the CLI) for the name of the keypair that was specified when the cluster was created.

After you have verified that you are using the correct key pair and that permissions are set correctly on the `.pem` file, you can use the following command to use SSH to connect to the master node, where you would replace `mykey.pem` with the name of your `.pem` file and `hadoop@ec2-01-001-001-1.compute-1.amazonaws.com` with the public DNS name of the master node (available through the `--describe` option in the CLI or through the Amazon EMR console.)

Important

You must use the login name `hadoop` when you connect to an Amazon EMR cluster node, otherwise an error similar to `Server refused our key` error may occur.

```
ssh -i mykey.pem hadoop@ec2-01-001-001-1.compute-1.amazonaws.com
```

For more information, see [Connect to the Master Node Using SSH \(p. 173\)](#).

If you are using IAM, do you have the proper Amazon EC2 policies set?

Because Amazon EMR uses EC2 instances as nodes, IAM users of Amazon EMR also need to have certain Amazon EC2 policies set in order for Amazon EMR to be able to manage those instances on the IAM user's behalf. If you do not have the required permissions set, Amazon EMR returns the error: **"User account is not authorized to call EC2."**

For more information about the Amazon EC2 policies your IAM account needs to set to run Amazon EMR, see [Amazon EMR Actions in User-Based IAM Policies \(p. 100\)](#).

Resource Errors

The following errors are commonly caused by constrained resources on the cluster.

Topics

- [Do you have enough HDFS space for your cluster? \(p. 225\)](#)
- [Are you seeing "EC2 Quota Exceeded" errors? \(p. 226\)](#)
- [Are you seeing "Too many fetch-failures" errors? \(p. 226\)](#)
- [Are you seeing "File could only be replicated to 0 nodes instead of 1" errors? \(p. 227\)](#)
- [Are your TaskTracker nodes being blacklisted? \(p. 227\)](#)

Do you have enough HDFS space for your cluster?

If you do not, Amazon EMR returns the following error: **"Cannot replicate block, only managed to replicate to zero nodes."** This error occurs when you generate more data in your cluster than can be stored in HDFS. You will see this error only while the cluster is running, because when the cluster ends it releases the HDFS space it was using.

The amount of HDFS space available to a cluster depends on the number and type of Amazon EC2 instances that are used as core nodes. All of the disk space on each Amazon EC2 instance is available to be used by HDFS. For more information about the amount of local storage for each EC2 instance type, see [Instance Types and Families](#) in the *Amazon EC2 User Guide for Linux Instances*.

The other factor that can affect the amount of HDFS space available is the replication factor, which is the number of copies of each data block that are stored in HDFS for redundancy. The replication factor increases with the number of nodes in the cluster: there are 3 copies of each data block for a cluster with 10 or more nodes, 2 copies of each block for a cluster with 4 to 9 nodes, and 1 copy (no redundancy) for clusters with 3 or fewer nodes. The total HDFS space available is divided by the replication factor. In some cases, such as increasing the number of nodes from 9 to 10, the increase in replication factor can actually cause the amount of available HDFS space to decrease.

For example, a cluster with ten core nodes of type `m1.large` would have 2833 GB of space available to HDFS $((10 \text{ nodes} \times 850 \text{ GB per node}) / \text{replication factor of } 3)$.

If your cluster exceeds the amount of space available to HDFS, you can add additional core nodes to your cluster or use data compression to create more HDFS space. If your cluster is one that can be stopped and restarted, you may consider using core nodes of a larger Amazon EC2 instance type. You might also consider adjusting the replication factor. Be aware, though, that decreasing the replication factor reduces the redundancy of HDFS data and your cluster's ability to recover from lost or corrupted HDFS blocks.

Are you seeing "EC2 Quota Exceeded" errors?

If you are getting messages that you are exceeding your Amazon EC2 instance quota, this may be for one of several reasons. Depending on configuration differences, it may take up to 5-20 minutes for previous clusters to terminate and release allocated resources. If you are getting an `EC2_QUOTA_EXCEEDED` error when you attempt to launch a cluster, it may be because resources from a recently terminated cluster have not yet been released. Furthermore, if you attempt to resize an instance group, you may also encounter this error when your new target size is greater than the current instance quota for the account. In these cases, you can either terminate and launch the cluster with a smaller target size. In all cases, you can terminate unused cluster resources or EC2 instances, [request that your Amazon EC2 quota be increased](#), or wait to re-launch a cluster.

Note

You cannot issue more than one resize request to the same cluster. Therefore, if your first request fails, you will have to potentially terminate your current cluster and launch another cluster with the number of instances desired.

Are you seeing "Too many fetch-failures" errors?

The presence of **"Too many fetch-failures"** or **"Error reading task output"** error messages in step or task attempt logs indicates the running task is dependent on the output of another task. This often occurs when a reduce task is queued to execute and requires the output of one or more map tasks and the output is not yet available.

There are several reasons the output may not be available:

- The prerequisite task is still processing. This is often a map task.
- The data may be unavailable due to poor network connectivity if the data is located on a different instance.
- If HDFS is used to retrieve the output, there may be an issue with HDFS.

The most common cause of this error is that the previous task is still processing. This is especially likely if the errors are occurring when the reduce tasks are first trying to run. You can check whether this is the case by reviewing the syslog log for the cluster step that is returning the error. If the syslog shows both map and reduce tasks making progress, this indicates that the reduce phase has started while there are map tasks that have not yet completed.

One thing to look for in the logs is a map progress percentage that goes to 100% and then drops back to a lower value. When the map percentage is at 100%, this does not mean that all map tasks are completed. It simply means that Hadoop is executing all the map tasks. If this value drops back below 100%, it means that a map task has failed and, depending on the configuration, Hadoop may try to reschedule the task. If the map percentage stays at 100% in the logs, look at the CloudWatch metrics, specifically `RunningMapTasks`, to check whether the map task is still processing. You can also find this information using the Hadoop web interface on the master node.

If you are seeing this issue, there are several things you can try:

- Instruct the reduce phase to wait longer before starting. You can do this by altering the Hadoop configuration setting `mapred.reduce.slowstart.completed.maps` to a longer time. For more information, see [\(Optional\) Create Bootstrap Actions to Install Additional Software](#) (p. 64).

- Match the reducer count to the total reducer capability of the cluster. You do this by adjusting the Hadoop configuration setting `mapred.reduce.tasks` for the job.
- Use a combiner class code to minimize the amount of outputs that need to be fetched.
- Check that there are no issues with the Amazon EC2 service that are affecting the network performance of the cluster. You can do this using the [Service Health Dashboard](#).
- Review the CPU and memory resources of the instances in your cluster to make sure that your data processing is not overwhelming the resources of your nodes. For more information, see [Configure Cluster Hardware and Networking \(p. 67\)](#).
- Check the version of the Amazon Machine Image (AMI) used in your Amazon EMR cluster. If the version is 2.3.0 through 2.4.4 inclusive, update to a later version. AMI versions in the specified range use a version of Jetty that may fail to deliver output from the map phase. The fetch error occurs when the reducers cannot obtain output from the map phase.

Jetty is an open-source HTTP server that is used for machine to machine communications within a Hadoop cluster.

Are you seeing "File could only be replicated to 0 nodes instead of 1" errors?

When a file is written to HDFS, it is replicated to multiple core nodes. When you see this error, it means that the NameNode daemon does not have any available DataNode instances to write data to in HDFS. In other words, block replication is not taking place. This error can be caused by a number of issues:

- The HDFS filesystem may have run out of space. This is the most likely cause.
- DataNode instances may not have been available when the job was run.
- DataNode instances may have been blocked from communication with the master node.
- Instances in the core instance group might not be available.
- Permissions may be missing. For example, the JobTracker daemon may not have permissions to create job tracker information.
- The reserved space setting for a DataNode instance may be insufficient. Check whether this is the case by checking the `dfs.datanode.du.reserved` configuration setting.

To check whether this issue is caused by HDFS running out of disk space, look at the `HDFSUtilization` metric in CloudWatch. If this value is too high, you can add additional core nodes to the cluster. Keep in mind that you can only add core nodes to a cluster, you cannot remove them. If you have a cluster that you think might run out of HDFS disk space, you can set an alarm in CloudWatch to alert you when the value of `HDFSUtilization` rises above a certain level. For more information, see [Manually Resizing a Running Cluster \(p. 199\)](#) and [Monitor Metrics with CloudWatch \(p. 157\)](#).

If HDFS running out of space was not the issue, check the DataNode logs, the NameNode logs and network connectivity for other issues that could have prevented HDFS from replicating data. For more information, see [View Log Files \(p. 146\)](#).

Are your TaskTracker nodes being blacklisted?

A TaskTracker node is a node in the cluster that accepts map and reduce tasks. These are assigned by a JobTracker daemon. The JobTracker monitors the TaskTracker node through a heartbeat.

There are a couple of situations in which the JobTracker daemon blacklists a TaskTracker node, removing it from the pool of nodes available to process tasks:

- If the TaskTracker node has not sent a heartbeat to the JobTracker daemon in the past 10 minutes (60000 milliseconds). This time period can be configured using the

`mapred.tasktracker.expiry.interval` configuration setting. For more information about changing Hadoop configuration settings, see [\(Optional\) Create Bootstrap Actions to Install Additional Software](#) (p. 64).

- If the TaskTracker node has more than 4 failed tasks. You can change this to a higher value using the `mapred.max.tracker.failures` configuration parameter. Other configuration settings you might want to change are the settings that control how many times to attempt a task before marking it as failed: `mapred.map.max.attempts` for map tasks and `mapreduce.reduce.maxattempts` for reduce tasks. For more information about changing Hadoop configuration settings, see [\(Optional\) Create Bootstrap Actions to Install Additional Software](#) (p. 64).

You can use the CloudWatch CLI to view the number of blacklisted TaskTracker nodes. The command for doing so is shown in the following example. For more information, see the [Amazon CloudWatch CLI Reference](#).

```
mon-get-stats NoOfBlackListedTaskTrackers --dimensions JobFlowId=JobFlowID --statistics  
Maximum --namespace AWS/ElasticMapReduce
```

The following example shows how to launch a cluster and use a bootstrap action to set the value of `mapred.max.tracker.failures` to 7, instead of the default 4.

Type the following command using the AWS CLI and replace *myKey* with the name of your EC2 key pair.

```
aws emr create-cluster --name "Test cluster" --ami-version 2.4 --applications Name=Hive  
Name=Pig \  
--use-default-roles --ec2-attributes KeyName=myKey \  
--instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m3.xlarge \  
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m3.xlarge \  
--bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/  
configure-hadoop,Name="Modified mapred.max.tracker.failures",Args=[ "-  
m", "mapred.max.tracker.failures=7" ]
```

Note

If you have not previously created the default EMR service role and EC2 instance profile, type `aws emr create-default-roles` to create them before typing the `create-cluster` subcommand.

When you launch a cluster using the preceding example, you can connect to the master node and see the changed configuration setting in `/home/hadoop/conf/mapred-site.xml`. The modified line will appear as shown in the following example.

```
<property><name>mapred.max.tracker.failures</name><value>7</value></property>
```

Streaming Cluster Errors

You can usually find the cause of a streaming error in a `syslog` file. Link to it on the **Steps** pane.

The following errors are common to streaming clusters.

Topics

- [Is data being sent to the mapper in the wrong format?](#) (p. 229)
- [Is your script timing out?](#) (p. 229)
- [Are you passing in invalid streaming arguments?](#) (p. 229)

- [Did your script exit with an error? \(p. 229\)](#)

Is data being sent to the mapper in the wrong format?

To check if this is the case, look for an error message in the `syslog` file of a failed task attempt in the task attempt logs. For more information, see [View Log Files \(p. 146\)](#).

Is your script timing out?

The default timeout for a mapper or reducer script is 600 seconds. If your script takes longer than this, the task attempt will fail. You can verify this is the case by checking the `syslog` file of a failed task attempt in the task attempt logs. For more information, see [View Log Files \(p. 146\)](#).

You can change the time limit by setting a new value for the `mapred.task.timeout` configuration setting. This setting specifies the number of milliseconds after which Amazon EMR will terminate a task that has not read input, written output, or updated its status string. You can update this value by passing an additional streaming argument `-jobconf mapred.task.timeout=800000`.

Are you passing in invalid streaming arguments?

Hadoop streaming supports only the following arguments. If you pass in arguments other than those listed below, the cluster will fail.

```
-blockAutoGenerateCacheFiles  
-cacheArchive  
-cacheFile  
-cmdenv  
-combiner  
-debug  
-input  
-inputformat  
-inputreader  
-jobconf  
-mapper  
-numReduceTasks  
-output  
-outputformat  
-partitioner  
-reducer  
-verbose
```

In addition, Hadoop streaming only recognizes arguments passed in using Java syntax; that is, preceded by a single hyphen. If you pass in arguments preceded by a double hyphen, the cluster will fail.

Did your script exit with an error?

If your mapper or reducer script exits with an error, you can locate the error in the `stderr` file of task attempt logs of the failed task attempt. For more information, see [View Log Files \(p. 146\)](#).

Custom JAR Cluster Errors

The following errors are common to custom JAR clusters.

Topics

- [Is your JAR throwing an exception before creating a job? \(p. 230\)](#)
- [Is your JAR throwing an error inside a map task? \(p. 230\)](#)

Is your JAR throwing an exception before creating a job?

If the main program of your custom JAR throws an exception while creating the Hadoop job, the best place to look is the `syslog` file of the step logs. For more information, see [View Log Files \(p. 146\)](#).

Is your JAR throwing an error inside a map task?

If your custom JAR and mapper throw an exception while processing input data, the best place to look is the `syslog` file of the task attempt logs. For more information, see [View Log Files \(p. 146\)](#).

Hive Cluster Errors

You can usually find the cause of a Hive error in the `syslog` file, which you link to from the **Steps** pane. If you can't determine the problem there, check in the Hadoop task attempt error message. Link to it on the **Task Attempts** pane.

The following errors are common to Hive clusters.

Topics

- [Are you using the latest version of Hive? \(p. 230\)](#)
- [Did you encounter a syntax error in the Hive script? \(p. 230\)](#)
- [Did a job fail when running interactively? \(p. 230\)](#)
- [Are you having trouble loading data to or from Amazon S3 into Hive? \(p. 230\)](#)

Are you using the latest version of Hive?

The latest version of Hive has all the current patches and bug fixes and may resolve your issue.

Did you encounter a syntax error in the Hive script?

If a step fails, look at the `stdout` file of the logs for the step that ran the Hive script. If the error is not there, look at the `syslog` file of the task attempt logs for the task attempt that failed. For more information, see [View Log Files \(p. 146\)](#).

Did a job fail when running interactively?

If you are running Hive interactively on the master node and the cluster failed, look at the `syslog` entries in the task attempt log for the failed task attempt. For more information, see [View Log Files \(p. 146\)](#).

Are you having trouble loading data to or from Amazon S3 into Hive?

If you are having trouble accessing data in Amazon S3, first check the possible causes listed in [Are you experiencing trouble loading data to or from Amazon S3? \(p. 223\)](#). If none of those issues is the cause, consider the following options specific to Hive.

- Make sure you are using the latest version of Hive, which has all the current patches and bug fixes that may resolve your issue. For more information, see [Apache Hive](#).

- Using `INSERT OVERWRITE` requires listing the contents of the Amazon S3 bucket or folder. This is an expensive operation. If possible, manually prune the path instead of having Hive list and delete the existing objects.
- Pre-cache the results of an Amazon S3 list operation locally on the cluster. You do this in HiveQL with the following command: `set hive.optimize.s3.query=true;`.
- Use static partitions where possible.
- In some versions of Hive and Amazon EMR, it is possible that using `ALTER TABLES` will fail because the table is stored in a different location than expected by Hive. The solution is to add or update following in `/home/hadoop/conf/core-site.xml`:

```
<property>
  <name>fs.s3n.endpoint</name>
  <value>s3.amazonaws.com</value>
</property>
```

VPC Errors

The following errors are common to VPC configuration in Amazon EMR.

Topics

- [Invalid Subnet Configuration \(p. 231\)](#)
- [Missing DHCP Options Set \(p. 231\)](#)
- [Permissions Errors \(p. 232\)](#)
- [Errors That Result in START_FAILED \(p. 232\)](#)
- [Cluster Terminated with errors and NameNode Fails to Start \(p. 233\)](#)

Invalid Subnet Configuration

On the **Cluster Details** page, in the **Status** field, you see an error similar to the following:

The subnet configuration was invalid: Cannot find route to InternetGateway in main RouteTable `rtb-id` for vpc `vpc-id`.

To solve this problem, you must create an Internet Gateway and attach it to your VPC. For more information, see [Adding an Internet Gateway to Your VPC](#).

Alternatively, verify that you have configured your VPC with **Enable DNS resolution** and **Enable DNS hostname support** enabled. For more information, see [Using DNS with Your VPC](#).

Missing DHCP Options Set

You see a step failure in the cluster system log (syslog) with an error similar to the following:

```
ERROR org.apache.hadoop.security.UserGroupInformation (main):
PrivilegedActionException as:hadoop (auth:SIMPLE) cause:java.io.IOException:
org.apache.hadoop.yarn.exceptions.ApplicationNotFoundException: Application with id
'application_id' doesn't exist in RM.
```

or

```
ERROR org.apache.hadoop.streaming.StreamJob (main): Error Launching job :
org.apache.hadoop.yarn.exceptions.ApplicationNotFoundException: Application with id
'application_id' doesn't exist in RM.
```

To solve this problem, you must configure a VPC that includes a DHCP Options Set whose parameters are set to the following values:

Note

If you use the AWS GovCloud (US) region, set domain-name to `us-gov-west-1.compute.internal` instead of the value used in the following example.

- **domain-name** = `ec2.internal`

Use `ec2.internal` if your region is US East (N. Virginia). For other regions, use `region-name.compute.internal`. For example in us-west-2, use **domain-name**=`us-west-2.compute.internal`.

- **domain-name-servers** = `AmazonProvidedDNS`

For more information, see [DHCP Options Sets](#).

Permissions Errors

A failure in the `stderr` log for a step indicates that an Amazon S3 resource does not have the appropriate permissions. This is a 403 error and the error looks like:

```
Exception in thread "main" com.amazonaws.services.s3.model.AmazonS3Exception: Access Denied
(Service: Amazon S3; Status Code: 403; Error Code: AccessDenied; Request ID: REQUEST_ID)
```

If the `ActionOnFailure` is set to `TERMINATE_JOB_FLOW`, then this would result in the cluster terminating with the state, `SHUTDOWN_COMPLETED_WITH_ERRORS`.

A few ways to troubleshoot this problem include:

- If you are using an Amazon S3 bucket policy within a VPC, make sure to give access to all buckets by creating a VPC endpoint and selecting **Allow all** under the Policy option when creating the endpoint.
- Make sure that any policies associated with S3 resources include the VPC in which you launch the cluster.
- Try running the following command from your cluster to verify you can access the bucket

```
hadoop fs -copyToLocal s3://path-to-bucket /tmp/
```

- You can get more specific debugging information by setting the `log4j.logger.org.apache.http.wire` parameter to `DEBUG` in `/home/hadoop/conf/log4j.properties` file on the cluster. You can check the `stderr` log file after trying to access the bucket from the cluster. The log file will provide more detailed information:

```
Access denied for getting the prefix for bucket - us-west-2.elasticmapreduce with path
samples/wordcount/input/
15/03/25 23:46:20 DEBUG http.wire: >> "GET /?prefix=samples%2Fwordcount%2Finput
%2F&delimiter=%2F&max-keys=1 HTTP/1.1[\r][\n]"
15/03/25 23:46:20 DEBUG http.wire: >> "Host: us-
west-2.elasticmapreduce.s3.amazonaws.com[\r][\n]"
```

Errors That Result in `START_FAILED`

Before AMI 3.7.0, for VPCs where a hostname is specified Amazon EC2 instances, Amazon EMR Amazon EMR maps the internal hostnames of the subnet with custom domain addresses as follows: `ip-X.X.X.X.customdomain.com.tld`. For example, if the hostname was `ip-10.0.0.10` and the VPC has the domain name option set to `customdomain.com`, the resulting hostname mapped by Amazon EMR would be `ip-10.0.1.0.customdomain.com`. An entry is added in `/etc/hosts` to resolve the hostname to

10.0.0.10. This behavior is changed with AMI 3.7.0 and now Amazon EMR honors the DHCP configuration of the VPC entirely. Previously, customers could also use a bootstrap action to specify a hostname mapping.

If you would like to preserve this behavior, you must provide the DNS and forward resolution setup you require for the custom domain.

Cluster Terminated with errors and NameNode Fails to Start

When launching an EMR cluster in a VPC which makes use of a custom DNS domain name, your cluster may fail with the following error message in the console:

```
Terminated with errors On the master instance(instance-id), bootstrap action 1 returned a non-zero return code
```

The failure is a result of the NameNode not being able to start up. This will result in the following error found in the NameNode logs, whose Amazon S3 URI is of the form: `s3://mybucket/logs/cluster-id/daemons/master instance-id/hadoop-hadoop-namenode-master node hostname.log.gz`:

```
2015-07-23 20:17:06,266 WARN
    org.apache.hadoop.hdfs.server.namenode.FSNamesystem (main): Encountered exception
    loading fsimage java.io.IOException: NameNode is not formatted.
    at

    org.apache.hadoop.hdfs.server.namenode.FSImage.recoverTransitionRead(FSImage.java:212)
    at

    org.apache.hadoop.hdfs.server.namenode.FSNamesystem.loadFSImage(FSNamesystem.java:1020)
    at

    org.apache.hadoop.hdfs.server.namenode.FSNamesystem.loadFromDisk(FSNamesystem.java:739)
    at
    org.apache.hadoop.hdfs.server.namenode.NameNode.loadNamesystem(NameNode.java:537)
    at
    org.apache.hadoop.hdfs.server.namenode.NameNode.initialize(NameNode.java:596)
    at org.apache.hadoop.hdfs.server.namenode.NameNode.<init>(NameNode.java:765)
    at
    org.apache.hadoop.hdfs.server.namenode.NameNode.<init>(NameNode.java:749)
    at
    org.apache.hadoop.hdfs.server.namenode.NameNode.createNameNode(NameNode.java:1441)
    at
    org.apache.hadoop.hdfs.server.namenode.NameNode.main(NameNode.java:1507)
```

This is due to a potential issue where an EC2 instance can have multiple sets of fully qualified domain names when launching EMR clusters in a VPC, which makes use of both an AWS-provided DNS server and a custom user-provided DNS server. If the user-provided DNS server does not provide any pointer (PTR) records for any A records used to designate nodes in an EMR cluster, clusters will fail starting up when configured in this way. The solution is to add 1 PTR record for every A record that is created when an EC2 instance is launched in any of the subnets in the VPC.

AWS GovCloud (US) Errors

The AWS GovCloud (US) region differs from other regions in its security, configuration, and default settings. As a result, use the following checklist to troubleshoot Amazon EMR errors that are specific to the AWS GovCloud (US) region before using more general troubleshooting recommendations.

- Verify that your IAM roles are correctly configured. For more information, see [Configure IAM Roles for Amazon EMR and Applications \(p. 106\)](#).

- Ensure that your VPC configuration has correctly configured DNS resolution/hostname support, Internet Gateway, and DHCP Option Set parameters. For more information, see [VPC Errors \(p. 231\)](#).

If these steps do not solve the problem, continue with the steps for troubleshooting common Amazon EMR errors. For more information, see [Common Errors in Amazon EMR \(p. 222\)](#).

Other Issues

Do you not see the cluster you expect in the Cluster List page or in results returned from ListClusters API?

Check the following:

- The cluster age is less than two months. Amazon EMR preserves metadata information about completed clusters for your reference, at no charge, for two months. The console does not provide a way to delete completed clusters from the console; these are automatically removed for you after two months.
- You have permissions to view the cluster. If the `visibleToAllUsers` property is set to false, other users in the same IAM account will not be able to view a cluster.
- You are viewing the correct region.

Write Applications that Launch and Manage Clusters

This documentation is for versions 4.x and 5.x of Amazon EMR. For information about Amazon EMR AMI versions 2.x and 3.x, see the [Amazon EMR Developer Guide \(PDF\)](#).

Topics

- [End-to-End Amazon EMR Java Source Code Sample \(p. 235\)](#)
- [Common Concepts for API Calls \(p. 238\)](#)
- [Use SDKs to Call Amazon EMR APIs \(p. 239\)](#)

You can access the functionality provided by the Amazon EMR API by calling wrapper functions in one of the AWS SDKs. The AWS SDKs provide language-specific functions that wrap the web service's API and simplify connecting to the web service, handling many of the connection details for you. For more information about calling Amazon EMR using one of the SDKs, see [Use SDKs to Call Amazon EMR APIs \(p. 239\)](#).

Important

The maximum request rate for Amazon EMR is one request every ten seconds.

End-to-End Amazon EMR Java Source Code Sample

Developers can call the Amazon EMR API using custom Java code to do the same things possible with the Amazon EMR console or CLI. This section provides the end-to-end steps necessary to install the AWS Toolkit for Eclipse and run a fully-functional Java source code sample that adds steps to an Amazon EMR cluster.

Note

This example focuses on Java, but Amazon EMR also supports several programming languages with a collection of Amazon EMR SDKs. For more information, see [Use SDKs to Call Amazon EMR APIs \(p. 239\)](#).

This Java source code example demonstrates how to perform the following tasks using the Amazon EMR API:

- Retrieve AWS credentials and send them to Amazon EMR to make API calls
- Configure a new custom step and a new predefined step
- Add new steps to an existing Amazon EMR cluster
- Retrieve cluster step IDs from a running cluster

Note

This sample demonstrates how to add steps to an existing cluster and thus requires that you have an active cluster on your account.

Before you begin, install a version of the **Eclipse IDE for Java EE Developers** that matches your computer platform. For more information, go to [Eclipse Downloads](#).

Next, install the Database Development plug-in for Eclipse.

To install the Database Development Eclipse plug-in

1. Open the Eclipse IDE.
2. Choose **Help** and **Install New Software**.
3. In the **Work with:** field, type `http://download.eclipse.org/releases/kepler` or the path that matches the version number of your Eclipse IDE.
4. In the items list, choose **Database Development** and **Finish**.
5. Restart Eclipse when prompted.

Next, install the Toolkit for Eclipse to make the helpful, pre-configured source code project templates available.

To install the Toolkit for Eclipse

1. Open the Eclipse IDE.
2. Choose **Help** and **Install New Software**.
3. In the **Work with:** field, type `https://aws.amazon.com//eclipse`.
4. In the items list, choose **AWS Toolkit for Eclipse** and **Finish**.
5. Restart Eclipse when prompted.

Next, create a new AWS Java project and run the sample Java source code.

To create a new AWS Java project

1. Open the Eclipse IDE.
2. Choose **File**, **New**, and **Other**.
3. In the **Select a wizard** dialog, choose **AWS Java Project** and **Next**.
4. In the **New AWS Java Project** dialog, in the **Project name:** field, enter the name of your new project, for example `EMR-sample-code`.
5. Choose **Configure AWS accounts...**, enter your public and private access keys, and choose **Finish**. For more information about creating access keys, see [How Do I Get Security Credentials?](#) in the *Amazon Web Services General Reference*.

Note

You should **not** embed access keys directly in code. The Amazon EMR SDK allows you to put access keys in known locations so that you do not have to keep them in code.

6. In the new Java project, right-click the **src** folder, then choose **New** and **Class**.
7. In the **Java Class** dialog, in the **Name** field, enter a name for your new class, for example **main**.
8. In the **Which method stubs would you like to create?** section, choose **public static void main(String[] args)** and **Finish**.
9. Enter the Java source code inside your new class and add the appropriate **import** statements for the classes and methods in the sample. For your convenience, the full source code listing is shown below.

Note

In the following sample code, replace the example cluster ID (`j-1HTE8WKS7SODR`) with a valid cluster ID in your account found either in the AWS Management Console or by using the following AWS CLI command:

```
aws emr list-clusters --active | grep "Id"
```

In addition, replace the example Amazon S3 path (`s3://mybucket/my-jar-location1`) with the valid path to your JAR. Lastly, replace the example class name (`com.my.Main1`) with the correct name of the class in your JAR, if applicable.

```
import java.io.IOException;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.PropertiesCredentials;
import com.amazonaws.services.elasticmapreduce.*;
import com.amazonaws.services.elasticmapreduce.model.AddJobFlowStepsRequest;
import com.amazonaws.services.elasticmapreduce.model.AddJobFlowStepsResult;
import com.amazonaws.services.elasticmapreduce.model.HadoopJarStepConfig;
import com.amazonaws.services.elasticmapreduce.model.StepConfig;
import com.amazonaws.services.elasticmapreduce.util.StepFactory;

public class main {

    public static void main(String[] args) {

        AWSCredentials credentials = null;
        try {
            credentials = new PropertiesCredentials(
                main.class.getResourceAsStream("AwsCredentials.properties"));
        } catch (IOException e1) {
            System.out.println("Credentials were not properly entered into
            AwsCredentials.properties.");
            System.out.println(e1.getMessage());
            System.exit(-1);
        }

        AmazonElasticMapReduce client = new AmazonElasticMapReduceClient(credentials);

        // predefined steps. See StepFactory for list of predefined steps
        StepConfig hive = new StepConfig("Hive", new StepFactory().newInstallHiveStep());

        // A custom step
        HadoopJarStepConfig hadoopConfig1 = new HadoopJarStepConfig()
            .withJar("s3://mybucket/my-jar-location1")
            .withMainClass("com.my.Main1") // optional main class, this can be omitted if
            jar above has a manifest
            .withArgs("--verbose"); // optional list of arguments
        StepConfig customStep = new StepConfig("Step1", hadoopConfig1);

        AddJobFlowStepsResult result = client.addJobFlowSteps(new AddJobFlowStepsRequest()
            .withJobFlowId("j-1HTE8WKS7SODR")
            .withSteps(hive, customStep));

        System.out.println(result.getStepIds());
    }
}
```

```
}  
}
```

10. Choose **Run**, **Run As**, and **Java Application**.
11. If the sample runs correctly, a list of IDs for the new steps appears in the Eclipse IDE console window. The correct output is similar to the following:

```
[s-39BLQZRB2E5E, s-1L6A4ZU2SAURC]
```

Common Concepts for API Calls

Topics

- [Endpoints for Amazon EMR \(p. 238\)](#)
- [Specifying Cluster Parameters in Amazon EMR \(p. 238\)](#)
- [Availability Zones in Amazon EMR \(p. 239\)](#)
- [How to Use Additional Files and Libraries in Amazon EMR Clusters \(p. 239\)](#)
- [Amazon EMR Sample Applications \(p. 239\)](#)

When you write an application that calls the Amazon EMR API, there are several concepts that apply when calling one of the wrapper functions of an SDK.

Endpoints for Amazon EMR

An endpoint is a URL that is the entry point for a web service. Every web service request must contain an endpoint. The endpoint specifies the AWS region where clusters are created, described, or terminated. It has the form `elasticmapreduce.regionname.amazonaws.com`. If you specify the general endpoint (`elasticmapreduce.amazonaws.com`), Amazon EMR directs your request to an endpoint in the default region. For accounts created on or after March 8, 2013, the default region is `us-west-2`; for older accounts, the default region is `us-east-1`.

For more information about the endpoints for Amazon EMR, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.

Specifying Cluster Parameters in Amazon EMR

The `Instances` parameters enable you to configure the type and number of EC2 instances to create nodes to process the data. Hadoop spreads the processing of the data across multiple cluster nodes. The master node is responsible for keeping track of the health of the core and task nodes and polling the nodes for job result status. The core and task nodes do the actual processing of the data. If you have a single-node cluster, the node serves as both the master and a core node.

The `KeepJobAlive` parameter in a `RunJobFlow` request determines whether to terminate the cluster when it runs out of cluster steps to execute. Set this value to `False` when you know that the cluster is running as expected. When you are troubleshooting the job flow and adding steps while the cluster execution is suspended, set the value to `True`. This reduces the amount of time and expense of uploading the results to Amazon Simple Storage Service (Amazon S3), only to repeat the process after modifying a step to restart the cluster.

If `KeepJobAlive` is `true`, after successfully getting the cluster to complete its work, you must send a `TerminateJobFlows` request or the cluster continues to run and generate AWS charges.

For more information about parameters that are unique to `RunJobFlow`, see [RunJobFlow](#). For more information about the generic parameters in the request, see [Common Request Parameters](#).

Availability Zones in Amazon EMR

Amazon EMR uses EC2 instances as nodes to process clusters. These EC2 instances have locations composed of Availability Zones and regions. Regions are dispersed and located in separate geographic areas. Availability Zones are distinct locations within a region insulated from failures in other Availability Zones. Each Availability Zone provides inexpensive, low-latency network connectivity to other Availability Zones in the same region. For a list of the regions and endpoints for Amazon EMR, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.

The `AvailabilityZone` parameter specifies the general location of the cluster. This parameter is optional and, in general, we discourage its use. When `AvailabilityZone` is not specified Amazon EMR automatically picks the best `AvailabilityZone` value for the cluster. You might find this parameter useful if you want to colocate your instances with other existing running instances, and your cluster needs to read or write data from those instances. For more information, see the [Amazon EC2 User Guide for Linux Instances](#).

How to Use Additional Files and Libraries in Amazon EMR Clusters

There are times when you might like to use additional files or custom libraries with your mapper or reducer applications. For example, you might like to use a library that converts a PDF file into plain text.

To cache a file for the mapper or reducer to use when using Hadoop streaming

- In the JAR args field, add the following argument:

```
-cacheFile s3://bucket/path_to_executable#local_path
```

The file, `local_path`, is in the working directory of the mapper, which could reference the file.

Amazon EMR Sample Applications

AWS provides tutorials that show you how to create complete applications, including:

- [Contextual Advertising using Apache Hive and Amazon EMR with High Performance Computing instances](#)
- [Parsing Logs with Apache Pig and Elastic MapReduce](#)
- [Finding Similar Items with Amazon EMR, Python, and Hadoop Streaming](#)
- [ItemSimilarity](#)
- [Word Count Example](#)

For more Amazon EMR code examples, go to [Sample Code & Libraries](#).

Use SDKs to Call Amazon EMR APIs

Topics

- [Using the AWS SDK for Java to Create an Amazon EMR Cluster \(p. 240\)](#)
- [Using the Java SDK to Sign an API Request \(p. 241\)](#)

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, see SDKs under [Tools for Amazon Web Services](#).

Using the AWS SDK for Java to Create an Amazon EMR Cluster

The AWS SDK for Java provides three packages with Amazon EMR functionality:

- [com.amazonaws.services.elasticmapreduce](#)
- [com.amazonaws.services.elasticmapreduce.model](#)
- [com.amazonaws.services.elasticmapreduce.util](#)

For more information about these packages, see the [AWS SDK for Java API Reference](#).

The following example illustrates how the SDKs can simplify programming with Amazon EMR. The code sample below uses the `StepFactory` object, a helper class for creating common Amazon EMR step types, to create an interactive Hive cluster with debugging enabled.

Note

If you are adding IAM user visibility to a new cluster, call [RunJobFlow](#) and set `VisibleToAllUsers=true`, otherwise IAM users cannot view the cluster.

```
AWSCredentials credentials = new BasicAWSCredentials(accessKey, secretKey);
AmazonElasticMapReduceClient emr = new AmazonElasticMapReduceClient(credentials);
String COMMAND_RUNNER = "command-runner.jar";
String DEBUGGING_COMMAND = "state-pusher-script";
String DEBUGGING_NAME = "Setup Hadoop Debugging";

StepFactory stepFactory = new StepFactory();

StepConfig enableddebugging = new StepConfig()
    .withName(DEBUGGING_NAME)
    .withActionOnFailure(ActionOnFailure.TERMINATE_CLUSTER)
    .withHadoopJarStep(new HadoopJarStepConfig()
        .withJar(COMMAND_RUNNER)
        .withArgs(DEBUGGING_COMMAND));

RunJobFlowRequest request = new RunJobFlowRequest()
    .withName("Hive Interactive")
    .withReleaseLabel("emr-4.1.0")
    .withSteps(enableddebugging)
    .withApplications(myApp)
    .withLogUri("s3://myawsbucket/")
    .withServiceRole("service_role")
    .withJobFlowRole("jobflow_role")
    .withInstances(new JobFlowInstancesConfig()
        .withEc2KeyName("keypair")
        .withInstanceCount(5)
        .withKeepJobFlowAliveWhenNoSteps(true)
        .withMasterInstanceType("m3.xlarge")
        .withSlaveInstanceType("m1.large"));

RunJobFlowResult result = emr.runJobFlow(request);
```

At minimum, you must pass a service role and jobflow role corresponding to EMR_DefaultRole and EMR_EC2_DefaultRole, respectively. You can do this by invoking this AWS CLI command for the same account. First, look to see if the roles already exist:

```
aws iam list-roles | grep EMR
```

Both the instance profile (EMR_EC2_DefaultRole) and the service role (EMR_DefaultRole) will be displayed if they exist:

```
"RoleName": "EMR_DefaultRole",  
  "Arn": "arn:aws:iam:::role/EMR_DefaultRole"  
"RoleName": "EMR_EC2_DefaultRole",  
  "Arn": "arn:aws:iam:::role/EMR_EC2_DefaultRole"
```

If the default roles do not exist, you can use the following AWS CLI command to create them:

```
aws emr create-default-roles
```

Using the Java SDK to Sign an API Request

Amazon EMR uses the Signature Version 4 signing process to construct signed requests to AWS. For more information, see [Signature Version 4 Signing Process](#) in the *Amazon Web Services General Reference*.