
Amazon EMR

Management Guide



Amazon EMR: Management Guide

Copyright © 2018 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	1
Submitting Work to a Cluster	2
Processing Data	2
Understanding the Cluster Lifecycle	3
Benefits	4
Cost Savings	4
AWS Integration	5
Deployment	5
Scalability and Flexibility	5
Reliability	6
Security	6
Monitoring	7
Management Interfaces	7
Architecture	8
Storage	8
Cluster Resource Management	8
Data Processing Frameworks	9
Applications and Programs	9
Getting Started	11
Step 1: Set Up Prerequisites	11
Sign Up for AWS	11
Create an Amazon S3 Bucket	12
Create an Amazon EC2 Key Pair	12
Step 2: Launch The Cluster	12
Launch the Sample Cluster	12
Summary of Quick Options	13
Step 3: Allow SSH Access	16
Step 4: Run a Hive Script to Process Data	17
Understanding The Data And Script	17
Submit the Hive Script as a Step	18
View the Results	19
Step 5: Clean Up Resources	20
Using EMR Notebooks	22
Considerations	22
Creating a Notebook	23
Creating Clusters for Notebooks	23
Notebook Limits per Cluster	24
Creating a Cluster When You Create a Notebook	24
Using an Existing Amazon EMR Cluster	24
Working with Notebooks	25
Understanding Notebook Status	25
Working with the Notebook Editor	26
Changing Clusters	26
Deleting Notebooks and Notebook Files	27
Sharing Notebook Files	27
Monitoring	28
Setting Up Spark User Impersonation	28
Using the Spark Job Monitoring Widget	29
Security	29
IAM Policy Actions for EMR Notebooks	30
Using Tags to Control User Permissions	31
Specifying EC2 Security Groups	36

Specifying the AWS Service Role	37
Plan and Configure Clusters	39
Configure Cluster Location and Data Storage	39
Choose an AWS Region	39
Work with Storage and File Systems	40
Prepare Input Data	43
Configure an Output Location	51
Use EMR File System (EMRFS)	55
Consistent View	56
Authorizing Access to EMRFS Data in Amazon S3	70
Specifying Amazon S3 Encryption Using EMRFS Properties	71
Control Cluster Termination	78
Configuring a Cluster to Auto-Terminate or Continue	79
Using Termination Protection	79
Working with AMIs	83
Using the Default AMI	84
Using a Custom AMI	85
Specifying the Amazon EBS Root Device Volume Size	90
Configure Cluster Software	91
Create Bootstrap Actions to Install Additional Software	92
Configure Cluster Hardware and Networking	96
Understand Node Types	96
Configure EC2 Instances	97
Configure Networking	102
Configure Instance Fleets or Instance Groups	111
Guidelines and Best Practices	122
Configure Cluster Logging and Debugging	126
Default Log Files	126
Archive Log Files to Amazon S3	127
Enable the Debugging Tool	129
Debugging Option Information	130
Tag Clusters	130
Tag Restrictions	131
Tag Resources for Billing	131
Add Tags to a New Cluster	132
Adding Tags to an Existing Cluster	132
View Tags on a Cluster	133
Remove Tags from a Cluster	134
Drivers and Third-Party Application Integration	134
Use Business Intelligence Tools with Amazon EMR	134
Security	136
Use IAM Policies to Allow and Deny User Permissions	137
Amazon EMR Actions in User-Based IAM Policies	137
Use Managed Policies for User Access	138
Use Inline Policies for User Permissions	140
Use Cluster Tagging with IAM Policies for Cluster-Specific Control	140
Use Kerberos Authentication	144
Supported Applications	144
Configure Kerberos	145
Configure a Cluster-Dedicated KDC	150
Configure a Cross-Realm Trust	152
Use an Amazon EC2 Key Pair for SSH Credentials	157
Encrypt Data in Transit and At Rest	157
Encryption Options	157
Create Keys and Certificates for Data Encryption	161
Configure IAM Roles for EMRFS	164
How IAM Roles for EMRFS Work	165

Set Up a Security Configuration with IAM Roles for EMRFS	165
Control Network Traffic with Security Groups	168
Working With Amazon EMR-Managed Security Groups	169
Working With Additional Security Groups	172
Specifying Security Groups	173
Use Security Configurations to Set Up Cluster Security	175
Create a Security Configuration	175
Specify a Security Configuration for a Cluster	186
Configure IAM Roles for Amazon EMR Permissions to AWS Services	187
EMR Role	187
EMR Role for EC2	187
Automatic Scaling Role	187
Service-Linked Role	188
Use Default IAM Roles and Managed Policies	188
Allow Users and Groups to Create and Modify Roles	194
Customize IAM Roles	194
Resource-Based Policies for AWS Glue	196
Use IAM Roles with Applications That Call AWS Services Directly	196
Using the Service-Linked Role	197
Manage Clusters	203
View and Monitor a Cluster	203
View Cluster Status and Details	203
Enhanced Step Debugging	208
View Application History	210
View Log Files	211
View Cluster Instances in Amazon EC2	215
CloudWatch Events and Metrics	216
View Cluster Application Metrics with Ganglia	236
Logging Amazon EMR API Calls in AWS CloudTrail	236
Connect to the Cluster	238
Connect to the Master Node Using SSH	239
View Web Interfaces Hosted on Amazon EMR Clusters	243
Terminate a Cluster	251
Terminate a Cluster Using the Console	252
Terminate a Cluster Using the AWS CLI	252
Terminate a Cluster Using the API	253
Scaling Cluster Resources	253
Using Automatic Scaling in Amazon EMR	254
Manually Resizing a Running Cluster	262
Cluster Scale-Down	268
Cloning a Cluster Using the Console	269
Submit Work to a Cluster	270
Work with Steps Using the CLI and Console	270
Submit Hadoop Jobs Interactively	272
Add More than 256 Steps to a Cluster	274
Automate Recurring Clusters with AWS Data Pipeline	274
Troubleshoot a Cluster	275
What Tools are Available for Troubleshooting?	275
Tools to Display Cluster Details	275
Tools to View Log Files	276
Tools to Monitor Cluster Performance	276
Viewing and Restarting Amazon EMR and Application Processes (Daemons)	276
Viewing Running Processes	277
Restarting Processes	277
Troubleshoot a Failed Cluster	278
Step 1: Gather Data About the Issue	278
Step 2: Check the Environment	279

Step 3: Look at the Last State Change	280
Step 4: Examine the Log Files	280
Step 5: Test the Cluster Step by Step	281
Troubleshoot a Slow Cluster	281
Step 1: Gather Data About the Issue	282
Step 2: Check the Environment	282
Step 3: Examine the Log Files	283
Step 4: Check Cluster and Instance Health	284
Step 5: Check for Arrested Groups	285
Step 6: Review Configuration Settings	286
Step 7: Examine Input Data	287
Common Errors in Amazon EMR	287
Input and Output Errors	288
Permissions Errors	290
Resource Errors	290
Streaming Cluster Errors	295
Custom JAR Cluster Errors	296
Hive Cluster Errors	297
VPC Errors	298
AWS GovCloud (US-West) Errors	300
Other Issues	301
Write Applications that Launch and Manage Clusters	302
End-to-End Amazon EMR Java Source Code Sample	302
Common Concepts for API Calls	304
Endpoints for Amazon EMR	305
Specifying Cluster Parameters in Amazon EMR	305
Availability Zones in Amazon EMR	305
How to Use Additional Files and Libraries in Amazon EMR Clusters	306
Use SDKs to Call Amazon EMR APIs	306
Using the AWS SDK for Java to Create an Amazon EMR Cluster	306
Using the Java SDK to Sign an API Request	307
AWS Glossary	308

What Is Amazon EMR?

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, in addition to this section:

- [Amazon EMR](#) – This service page provides the Amazon EMR highlights, product details, and pricing information.
- [Getting Started: Analyzing Big Data with Amazon EMR \(p. 11\)](#) – These tutorials get you started using Amazon EMR quickly.

In This Section

- [Overview of Amazon EMR \(p. 1\)](#)
- [Benefits of Using Amazon EMR \(p. 4\)](#)
- [Overview of Amazon EMR Architecture \(p. 8\)](#)

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 \(p. 1\)](#)
- [Submitting Work to a Cluster \(p. 2\)](#)
- [Processing Data \(p. 2\)](#)
- [Understanding the Cluster Lifecycle \(p. 3\)](#)

Understanding Clusters and Nodes

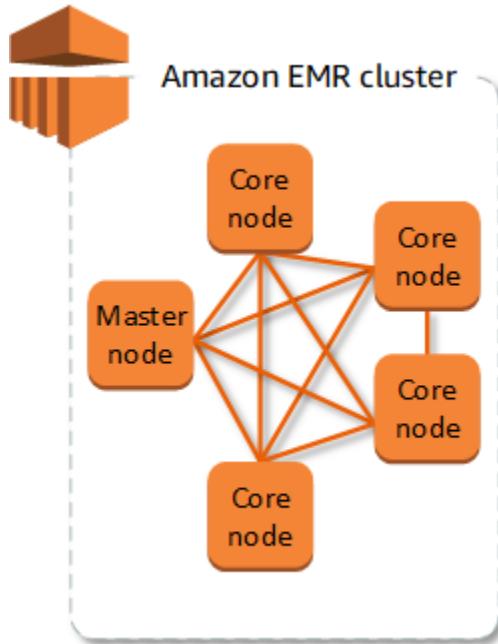
The central component of Amazon EMR is the *cluster*. A cluster is a collection of Amazon Elastic Compute Cloud (Amazon EC2) instances. Each instance in the cluster is called a *node*. Each node has a role within the cluster, referred to as the *node type*. Amazon EMR also installs different software 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 to coordinate the distribution of data and tasks among other nodes for processing. The master node tracks the status of tasks and monitors the health of the cluster. Every cluster has a master node, and it's possible to create a single-node cluster with only the master node.
- **Core node:** A node with software components that run tasks and store data in the Hadoop Distributed File System (HDFS) on your cluster. Multi-node clusters have at least one core node.

- **Task node:** A node with software components that only runs tasks and does not store data in HDFS. Task nodes are optional.

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



Submitting Work to a Cluster

When you run a 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 functions that you specify as steps when you create a cluster. This is typically done for clusters that process a set amount of data and then terminate when processing is complete.
- 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 jobs. For more information, see [Submit Work to a Cluster \(p. 270\)](#).
- Create a cluster, connect to the master node and other nodes as required using SSH, and use the interfaces that the installed applications provide to perform tasks and submit queries, either scripted or interactively. For more information, see the [Amazon EMR Release Guide](#).

Processing Data

When you launch your cluster, you choose the frameworks and applications to install for your data processing needs. To process data in your Amazon EMR cluster, you can submit jobs or queries directly to installed applications, or you can run *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 \(p. 238\)](#).

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:

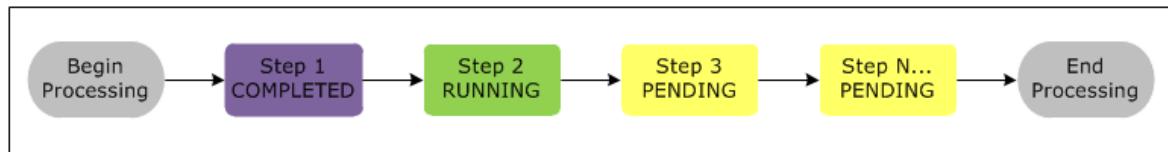
1. Submit an input dataset for processing.
2. Process the output of the first step by using a Pig program.
3. Process a second input dataset by using a Hive program.
4. Write an output dataset.

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:

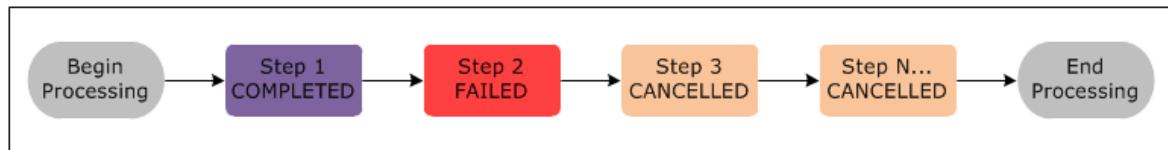
1. A request is submitted to begin processing steps.
2. The state of all steps is set to **PENDING**.
3. When the first step in the sequence starts, its state changes to **RUNNING**. The other steps remain in the **PENDING** state.
4. After the first step completes, its state changes to **COMPLETED**.
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 **TERMINATED_WITH_ERRORS**. You can determine what happens next for each step. By default, any remaining steps in the sequence are set to **CANCELLED** and do not run. You can also choose to ignore the failure and allow remaining steps to proceed, or to terminate the cluster immediately.

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 EC2 instances in the cluster for each instance according to your specifications. For more information, see [Configure Cluster Hardware and Networking \(p. 96\)](#). For all instances, Amazon EMR uses the default AMI for Amazon EMR or a custom Amazon Linux AMI that you specify. For more information, see [Using a Custom AMI \(p. 85\)](#). During this phase, the cluster state is **STARTING**.
2. Amazon EMR runs *bootstrap actions* that you specify on each instance. You can use bootstrap actions to install custom applications and perform customizations that you require. For more information, see [Create Bootstrap Actions to Install Additional Software \(p. 92\)](#). During this phase, the cluster state is **BOOTSTRAPPING**.
3. Amazon EMR installs the native applications that you specify when you create the cluster, such as Hive, Hadoop, Spark, and so on.
4. After bootstrap actions are successfully completed and native applications are installed, the cluster state is **RUNNING**. At this point, you can connect to cluster instances, and the cluster sequentially runs any steps that you specified when you created the cluster. You can submit additional steps, which run after any previous steps complete. For more information, see [Work with Steps Using the CLI and Console \(p. 270\)](#).
5. After steps run successfully, the cluster goes into a **WAITING** state. If a cluster is configured to auto-terminate after the last step is complete, it goes into a **SHUTTING_DOWN** state.
6. After all instances are terminated, the cluster goes into the **COMPLETED** state.

A failure during the cluster lifecycle causes Amazon EMR to terminate the cluster and all of its instances unless you enable termination protection. If a cluster terminates because of a failure, any data stored on the cluster is deleted, and the cluster state is set to **FAILED**. If you enabled termination protection, you can retrieve data from your cluster, and then remove termination protection and terminate the cluster. For more information, see [Using Termination Protection \(p. 79\)](#).

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 \(p. 4\)](#)
- [AWS Integration \(p. 5\)](#)
- [Deployment \(p. 5\)](#)
- [Scalability and Flexibility \(p. 5\)](#)
- [Reliability \(p. 6\)](#)
- [Security \(p. 6\)](#)
- [Monitoring \(p. 7\)](#)
- [Management Interfaces \(p. 7\)](#)

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 rates, but you can reduce the cost even further by purchasing Reserved Instances or 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.

For more information about pricing options and details, see [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 \(p. 96\)](#).

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 \(p. 91\)](#).

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 \(p. 262\)](#).

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? \(p. 123\)](#).

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 \(p. 40\)](#).

Reliability

Amazon EMR monitors nodes in your cluster and automatically terminates and replaces an instance in case of 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 *long-running clusters*.

Additionally, you can configure termination protection to prevent 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 before termination. The default settings for these options differ depending on whether you launch your cluster by using the console, CLI, or API. For more information, see [Using Termination Protection \(p. 79\)](#).

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 [Use IAM Policies to Allow and Deny User Permissions \(p. 137\)](#) and [Amazon EMR Actions in User-Based IAM Policies \(p. 137\)](#).

Additionally, Amazon EMR uses IAM roles for the Amazon EMR service itself and the EC2 instance profile for the instances. These roles grant permissions 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 Permissions to AWS Services \(p. 187\)](#).

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 for more advanced rules. For more information, see [Control Network Traffic with Security Groups \(p. 168\)](#).

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 [Configure Networking \(p. 102\)](#).

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 track 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 \(p. 236\)](#).

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 or use Kerberos for authentication. If you use SSH, an Amazon EC2 key pair is required. For more information, see [Use an Amazon EC2 Key Pair for SSH Credentials \(p. 157\)](#).

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 \(p. 126\)](#).

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 \(p. 223\)](#).

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. If you prefer working from a command line, using the AWS CLI is the best option. For more information, see [Amazon EMR in the AWS CLI Command 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 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 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 \(p. 8\)](#)
- [Cluster Resource Management \(p. 8\)](#)
- [Data Processing Frameworks \(p. 9\)](#)
- [Applications and Programs \(p. 9\)](#)

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 that 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 pre-attached 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 that administers YARN components, keeps the cluster healthy, and communicates with Amazon EMR.

Because Spot Instances are often used to run task nodes, Amazon EMR has default functionality for scheduling YARN jobs so that running jobs don't fail when task nodes running on Spot Instances are terminated. Amazon EMR does this by allowing application master processes to run only on core nodes. The application master process controls running jobs and needs to stay alive for the life of the job.

Amazon EMR release version 5.19.0 and later uses the built-in [YARN node labels](#) feature to achieve this. (Earlier versions used a code patch). Properties in the `yarn-site` and `capacity-scheduler` configuration classifications are configured by default so that the YARN capacity-scheduler and fair-scheduler take advantage of node labels. Amazon EMR automatically labels core nodes with the `CORE` label, and sets properties so that application masters are scheduled only on nodes with the `CORE` label. Manually modifying related properties in the `yarn-site` and `capacity-scheduler` configuration classifications, or directly in associated XML files, could break this feature or modify this functionality.

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 key-value 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 generates 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 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 tutorial walks you through the process of creating a sample Amazon EMR cluster using **Quick Create** options in the AWS Management Console. After you create the cluster, you submit a Hive script as a *step* to process sample data stored in Amazon Simple Storage Service (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 set up a cluster for evaluation purposes as quickly as possible. If you have questions or get stuck, reach out to the Amazon EMR team by posting on our [Discussion Forum](#).

The sample cluster that you create runs in a live environment. Charges accrue for cluster instances at the per-second rate for Amazon EMR pricing. For more information, see [Amazon EMR Pricing](#). These charges vary by region. The cost should be minimal because the cluster should run for less than an hour after the cluster is provisioned.

Charges might also accrue for the storage of query output files that you store in Amazon S3 as part of the tutorial. The file is small, so charges should be minimal. Also, 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 the usage limits of the AWS Free Tier. For more information, see [Amazon S3 Pricing](#) and [AWS Free Tier](#).

If you are considering Amazon EMR in a production capacity, use the [AWS Simple Monthly Calculator](#) to estimate your costs.

Steps in This Tutorial

- [Step 1: Set Up Prerequisites for Your Sample Cluster \(p. 11\)](#)
- [Step 2: Launch Your Sample Amazon EMR Cluster \(p. 12\)](#)
- [Step 3: Allow SSH Connections to the Cluster From Your Client \(p. 16\)](#)
- [Step 4: Process Data By Running The Hive Script as a Step \(p. 17\)](#)
- [Step 5: Terminate the Cluster and Delete the Bucket \(p. 20\)](#)

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.

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 then choose **Create an AWS Account**.

Note

If you previously signed in to the AWS Management Console using AWS account root user credentials, choose **Sign in to a different account**. If you previously signed in to the console using IAM credentials, choose **Sign-in using root account credentials**. Then choose **Create a new AWS account**.

2. Follow the online instructions.

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

Create an Amazon S3 Bucket

In this tutorial, you specify an Amazon S3 bucket and folder to store the output data from a Hive query. The tutorial uses the default log location, but you can also specify a custom location if you prefer. Because of Hadoop requirements, bucket and folder names that you use with Amazon EMR have the following limitations:

- They must contain only lowercase letters, numbers, periods (.), and hyphens (-).
- They cannot end in numbers.

If you already have access to a folder that meets these requirements, you can use it for this tutorial. The output folder should be empty. Another requirement to remember is that bucket names must be unique *across all AWS accounts*.

For more information about creating a bucket, see [Create a Bucket](#) in the *Amazon Simple Storage Service Getting Started Guide*. After you create the bucket, choose it from the list and then choose **Create folder**, replace **New folder** with a name that meets the requirements, and then choose **Save**.

The bucket and folder name used later in the tutorial is `s3://mybucket/MyHiveQueryResults`.

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*. Use this procedure for Mac OS as well.

Step 2: Launch Your Sample Amazon EMR Cluster

In this step, you launch your sample cluster by using **Quick Options** in the Amazon EMR console and leaving most options to their default values. To learn more about these options, see [Summary of Quick Options \(p. 13\)](#) after the procedure. You can also select **Go to advanced options** to explore the additional configuration options available for a cluster. Before you create your cluster for this tutorial, make sure that you meet the requirements in [Step 1: Set Up Prerequisites for Your Sample Cluster \(p. 11\)](#).

Launch the Sample Cluster

To launch the sample Amazon EMR cluster

1. Sign in to the AWS Management Console and open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**.
3. On the **Create Cluster - Quick Options** page, accept the default values except for the following fields:

- Enter a **Cluster name** that helps you identify the cluster, for example, *My First EMR Cluster*.
 - Under **Security and access**, choose the **EC2 key pair** that you created in [Create an Amazon EC2 Key Pair \(p. 12\)](#).
4. Choose **Create cluster**.

The cluster status page with the cluster **Summary** appears. You can use this page to monitor the progress of cluster creation and view details about cluster status. As cluster creation tasks finish, items on the status page update. You may need to choose the refresh icon on the right or refresh your browser to receive updates.

Under **Network and hardware**, find the **Master** and **Core** instance status. The status goes from **Provisioning** to **Bootstrapping** to **Waiting** during the cluster creation process. For more information, see [Understanding the Cluster Lifecycle \(p. 3\)](#).

As soon as you see the links for **Security groups for Master** and **Security Groups for Core & Task**, you can move on to the next step, but you may want to wait until the cluster starts successfully and is in the **Waiting** state.

For more information about reading the cluster summary, see [View Cluster Status and Details \(p. 203\)](#).

Summary of Quick Options

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	When logging is enabled, Amazon EMR writes detailed log data to the Amazon S3 folder specified. Logging can only be enabled when you create the cluster and the setting can't be changed later. A default Amazon S3 bucket is specified. You can optionally specify your own. For more information, see View Log Files Archived to Amazon S3 (p. 213) .
S3 folder	s3://aws-logs-<i>account_number-region</i>/elasticmapreduce/	This option specifies the path to a folder in an Amazon S3 bucket where you want Amazon EMR to write log data. If the default 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 an Amazon S3 folder.

Console field	Default value	Description
Launch mode	Cluster	<p>This option specifies whether to launch a long-running cluster or a cluster that terminates after running any steps that you specify.</p> <p>With the Cluster option, the cluster continues to run until you terminate it, which is called a <i>long-running cluster</i>. If you choose Step execution, Amazon EMR prompts you to add and configure steps. You can use steps to submit work to a cluster. After the steps that you specify finish executing, the cluster terminates automatically. For more information, see Configuring a Cluster to Auto-Terminate or Continue (p. 79).</p>
Release	<i>emr-5.20.0</i>	<p>This option specifies the Amazon EMR release version to use when the cluster is created. The Amazon EMR release determines the version of open-source applications, such as Hadoop and Hive, that Amazon EMR installs. The label for the latest release version is selected by default. You can select an earlier Amazon EMR release if you need different versions of open-source applications for compatibility with your solution. Some Amazon EMR features and applications may not be available when using earlier Amazon EMR release versions, so recommend that you use the latest release version whenever possible. For more information about each Amazon EMR release version, see Amazon EMR Release Guide.</p>

Console field	Default value	Description
Applications	Core Hadoop	<p>This option determines the open-source applications from the big data ecosystem to install on your cluster. The most common application combinations are available using quick start. To select your own combination of applications, including additional applications not listed in quick start, choose Go to advanced options. For information about the applications and versions available with each Amazon EMR release version, see Amazon EMR Release Guide.</p> <p>In addition, if an application isn't available for Amazon EMR to install, or you need to install a custom application on all cluster instances, you can use a <i>bootstrap action</i>. For more information, see Create Bootstrap Actions to Install Additional Software (p. 92). If you select Step execution, Amazon EMR chooses the applications to install based on what your steps require.</p>
Instance type	m4.large	<p>This option determines the Amazon EC2 instance type that Amazon EMR initializes for the instances that run in your cluster. The default instance selection varies by region and some instance types may not be available in some regions. For more information, see Configure Cluster Hardware and Networking (p. 96).</p>
Number of instances	3	<p>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, which is the master node. For guidance about choosing instance types and the number of instances, see Cluster Configuration Guidelines and Best Practices (p. 122).</p>

Console field	Default value	Description
EC2 key pair	Choose an option	This specifies the Amazon EC2 key pair to use when connecting to the nodes in your cluster over a Secure Shell (SSH) connection. We strongly recommend that you create and specify an Amazon EC2 key pair. If you do not select a key pair, you cannot connect to the cluster to submit steps or interact with applications. For more information, see Connect to the Cluster (p. 238) . To connect, you also need to create an inbound rule in the security group to allow SSH connections.
Permissions	Default	Use this option to specify the AWS Identity and Access Management roles that the cluster uses. These roles determine the permissions that Amazon EMR and the applications running on cluster instances have to interact with other AWS services. You can choose Custom to specify your own roles. We recommend using the default roles to start with. For more information, see Configure IAM Roles for Amazon EMR Permissions to AWS Services (p. 187) .

Step 3: Allow SSH Connections to the Cluster From Your Client

Security groups act as virtual firewalls to control inbound and outbound traffic to your cluster. The default Amazon EMR-managed security groups associated with cluster instances do not allow inbound SSH connections as a security precaution. To connect to cluster nodes using SSH so that you can use the command line and view web interfaces that are hosted on the cluster, you need to add inbound rules that allow SSH traffic from trusted clients. Allowing SSH from your client isn't a requirement to complete the tutorial—the Hive script runs and you can access the results in Amazon S3—but accessing clusters using SSH is a common requirement. Performing this step allows you to explore next steps with your new cluster after you complete the tutorial. For more information about security groups, see [Control Network Traffic with Security Groups \(p. 168\)](#) and [Security Groups for Your VPC](#) in the *Amazon VPC User Guide*.

To allow inbound SSH traffic to cluster EC2 instances

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

3. Choose the **Name** of the cluster.
 4. Under **Security and access** choose the **Security groups for Master** link.
 5. Choose **ElasticMapReduce-master** from the list.
 6. Choose **Inbound**, **Edit** to add a new inbound rule.
 7. Scroll down through the listing of default rules and choose **Add Rule** at the bottom of the list.
 8. For **Type**, select **SSH**. For source, select **My IP**.
- This automatically adds the IP address of your client computer as the source address. Alternatively, you can add a range of **Custom** trusted client IP addresses and choose **Add rule** to create additional rules for other clients. In many network environments, IP addresses are allocated dynamically, so you may need to periodically edit security group rules to update the IP address of trusted clients.
9. Choose **Save**.
 10. Choose **ElasticMapReduce-slave** from the list and repeat the steps above to allow SSH client access to core and task nodes from trusted clients.

Step 4: Process Data By Running The Hive Script as a Step

With your cluster up and running, you can now submit a Hive script. In this tutorial, you submit the Hive script as a step using the Amazon EMR console. In Amazon EMR, a *step* is a unit of work that contains one or more jobs. As you learned in [Step 2: Launch Your Sample Amazon EMR Cluster \(p. 12\)](#), you can submit steps to a long-running cluster, which is what we do in this step. You can also specify steps when you create a cluster, or you could connect to the master node, create the script in the local file system, and run it using the command line, for example `hive -f Hive_CloudFront.q`.

Understanding The Data And Script

The sample data and script that you use in this tutorial are already available in an Amazon S3 location that you can access.

The sample data is a series of Amazon CloudFront access log files. For more information about CloudFront and log file formats, see [Amazon CloudFront Developer Guide](#). The data is stored in Amazon S3 at `s3://region.elasticmapreduce.samples/cloudfront/data` where `region` is your region, for example, `us-west-2`. When you enter the location when you submit the step, you omit the `cloudfront/data` portion because the script adds it.

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

```
2014-07-05 20:00:00 LHR3 4260 10.0.0.15 GET eabcd12345678.cloudfront.net /test-image-1.jpeg 200 - Mozilla/5.0%20(MacOS;%20;%20Windows%20NT%205.1;%20en-US;%20rv:1.9.0.9)%20Gecko/2009040821%20IE/3.0.9
```

The sample script calculates the total number of requests per operating system over a specified time frame. The script uses HiveQL, which is a SQL-like scripting language for data warehousing and analysis. The script is stored in Amazon S3 at `s3://region.elasticmapreduce.samples/cloudfront/code/Hive_CloudFront.q` where `region` is your region.

The sample Hive script does the following:

- Creates a Hive table schema named `cloudfront_logs`. For more information about Hive tables, see the [Hive Tutorial](#) on the Hive wiki.

- Uses the built-in regular expression serializer/deserializer (RegEx SerDe) to parse the input data and apply the table schema. For more information, see [SerDe](#) on the Hive wiki.
 - Runs a HiveQL query against the `cloudfront_logs` table and writes the query results to the Amazon S3 output location that you specify.

The contents of the `Hive_CloudFront.q` script are shown below. The `$(INPUT)` and `$(OUTPUT)` variables are replaced by the Amazon S3 locations that you specify when you submit the script as a step. When you reference data in Amazon S3 as this script does, Amazon EMR uses the EMR File System (EMRFS) to read input data and write output data.

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 have been uploaded to Amazon S3, and you specify the output location as the folder you created earlier in [Create an Amazon S3 Bucket \(p. 12\)](#).

To run the Hive script by submitting it 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. Make sure the cluster is in a **Waiting** state.
 3. Choose **Steps**, and then choose **Add step**.
 4. Configure the step according to the following guidelines:
 - For **Step type**, choose **Hive program**.
 - For **Name**, you can leave the default or type a new name. If you have many steps in a cluster, the name helps you keep track of them.
 - For **Script S3 location**, type `s3://region.elasticmapreduce.samples/cloudfront/code/Hive_CloudFront.q`. Replace `region` with your region identifier. For example, `s3://us-west-2.elasticmapreduce.samples/cloudfront/code/Hive_CloudFront.q` if you are

working in the **Oregon** region. For a list of regions and corresponding Region identifiers, see [AWS Regions and Endpoints for Amazon EMR](#) in the *AWS General Reference*.

- For **Input S3 location**, type `s3://region.elasticmapreduce.samples`

Replace `region` with your region identifier.

- For **Output S3 location**, type or browse to the output bucket that you created in [Create an Amazon S3 Bucket \(p. 12\)](#).

- For **Action on failure**, accept the default option **Continue**. This specifies that if the step fails, the cluster continues to run and processes subsequent steps. The **Cancel and wait** option specifies that a failed step should be canceled, that subsequent steps should not run, and that the cluster should continue running. The **Terminate cluster** option specifies that the cluster should terminate if the step fails.

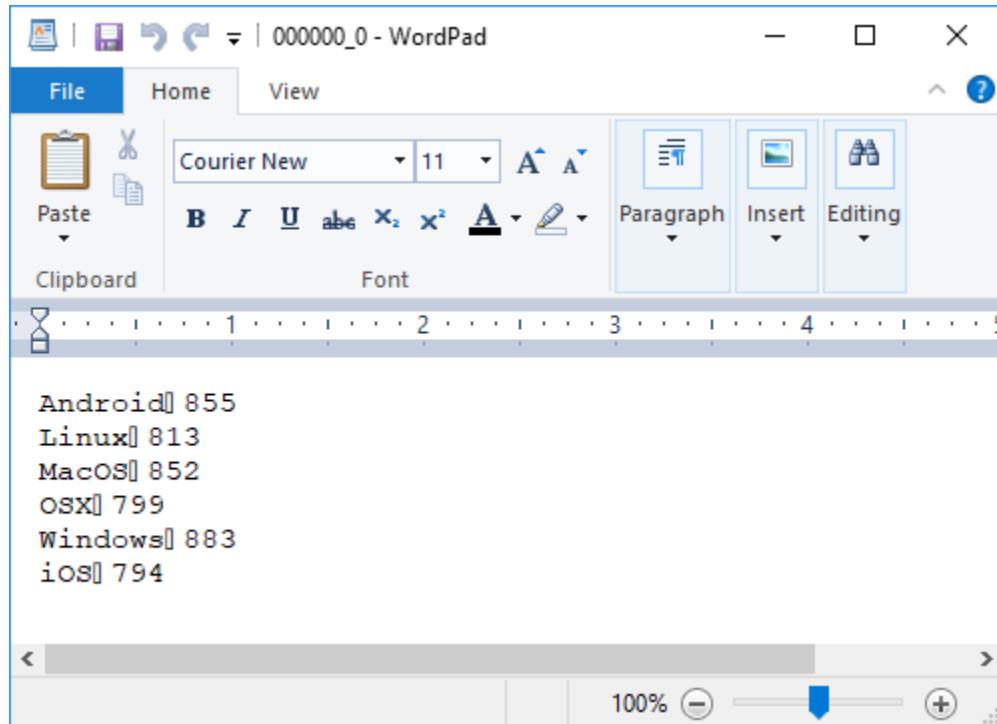
5. Choose **Add**. The step appears in the console with a status of **Pending**.
6. The status of the step changes from **Pending** to **Running** to **Completed** as the step runs. To update the status, choose the refresh icon to the right of the **Filter**. The script takes approximately a minute to run.

View the Results

After the step completes successfully, the Hive query output is saved as a text file 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. Choose the **Bucket name** and then the folder that you set up earlier. For example, `mybucket` and then `MyHiveQueryResults`.
3. The query writes results to a folder within your output folder named `os_requests`. Choose that folder. There should be a single file named `000000_0` in the folder. This is a text file that contains your Hive query results.
4. Choose the file, and then choose **Download** to save it locally.
5. Use the text editor that you prefer to open the file. The output file shows the number of access requests ordered by operating system. The following example shows the output in WordPad:



Step 5: Terminate the Cluster and Delete the Bucket

After you complete the tutorial, you may want to terminate your cluster and delete your Amazon S3 bucket to avoid additional charges.

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 terminated clusters so that they aren't viewable in the console. Terminated clusters are removed from the cluster when the metadata is removed.

To terminate the cluster

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Clusters**, choose your cluster, and then choose **Terminate**.

Clusters are often created with termination protection on, which helps prevent accidental shutdown. If you followed the tutorial precisely, termination protection should be off. If termination protection is on, you are prompted to change the setting as a precaution before terminating the cluster. Choose **Change, Off**.

To delete the output bucket

1. Open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. Choose the bucket from the list, so that the whole bucket row is selected.
3. Choose **delete bucket**, type the name of the bucket, and then click **Confirm**.

For more information about deleting folders and buckets, go to [How Do I Delete an S3 Bucket](#) in the *Amazon Simple Storage Service Getting Started Guide*.

Using Amazon EMR Notebooks

Use Amazon EMR Notebooks to create and open Jupyter notebooks with the Amazon EMR console. You can use an EMR notebook with Amazon EMR clusters running [Apache Spark](#) to remotely run queries and code. An EMR notebook is a "serverless" Jupyter notebook. Unlike a traditional notebook, the contents of an EMR notebook itself—the equations, visualizations, queries, models, code, and narrative text—are saved in Amazon S3 separately from the cluster that runs the code. This provides an EMR notebook with durable storage, efficient access, and flexibility.

An Amazon EMR cluster is required to execute the code and queries within an EMR notebook, but the notebook isn't locked to the cluster. This makes support for transient clusters more efficient. You can start a cluster, attach an EMR notebook to it, and terminate the cluster. The notebook still exists, so the next time you want to analyze or model data, you can create another cluster and attach the same notebook to it.

You can also stop an EMR notebook attached to a running cluster and then change clusters. You can attach to another running cluster or create a new one without having to reconfigure the notebook or terminate clusters. These features let you run clusters on-demand to save cost. Also, you can save time re-configuring notebooks when you want to use the same notebook on a different cluster or dataset.

Applicable charges for Amazon S3 storage and for Amazon EMR clusters apply.

Topics

- [Considerations When Using EMR Notebooks \(p. 22\)](#)
- [Creating a Notebook \(p. 23\)](#)
- [Creating Amazon EMR Clusters for Notebooks \(p. 23\)](#)
- [Working with Notebooks \(p. 25\)](#)
- [Monitoring Spark User and Job Activity \(p. 28\)](#)
- [EMR Notebooks Security and Access Control \(p. 29\)](#)

Considerations When Using EMR Notebooks

EMR Notebooks runs [Jupyter Notebook version 5.7.0](#) and Python 3.6.5.

EMR Notebooks is pre-configured with the following kernels and library packages installed.

Kernels

- PySpark
- PySpark3
- Python3
- Spark
- SparkR

Library Packages

- [Packages for 64-bit Linux with Python 3.6](#)

Installing additional library packages from within the notebook editor is not currently supported. If you require customized kernels or library packages, install them using bootstrap actions or by specifying a custom Amazon Linux AMI for Amazon EMR when you create a cluster. For more information, see [Create Bootstrap Actions to Install Additional Software \(p. 92\)](#) and [Using a Custom AMI \(p. 85\)](#).

Clusters that you use with an EMR notebook have requirements. The number of notebooks that you can use with a cluster is limited by the master node configuration. For more information, see [Creating Amazon EMR Clusters for Notebooks \(p. 23\)](#).

Creating a Notebook

You create an EMR notebook using the Amazon EMR console. Creating notebooks using the AWS CLI or the Amazon EMR API is not supported.

To create an EMR notebook

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Notebooks**, **Create notebook**.
3. Enter a **Notebook name** and an optional **Notebook description**.
4. If you have an active cluster running Hadoop, Spark, and Livy to which you want to attach the notebook, leave the default, select **Choose**, select a cluster from the list, and then **Choose cluster**. Only clusters that meet the requirements are listed.

—or—

Choose **Create a cluster**, enter a **Cluster name** and choose the number and type of EC2 instances for the cluster. One instance hosts the master node, and the remainder are used for core nodes. You can also choose a custom service role and EC2 instance profile, if necessary. For more information, see [Creating a Cluster When You Create a Notebook \(p. 24\)](#).

5. For **Security groups**, choose **Use default security groups**. Alternatively, choose **Choose security groups** and select custom security groups. You select one for the master instance and another for the notebook service. For more information, see [the section called "Specifying EC2 Security Groups" \(p. 36\)](#).
6. For **AWS Service Role**, leave the default or choose a custom role from the list. For more information, see [Specifying the AWS Service Role \(p. 37\)](#).
7. For **Notebook location** choose the location in Amazon S3 where the notebook file is saved, or specify your own location. If the bucket and folder don't exist, Amazon EMR creates it.

Amazon EMR creates a folder with the **Notebook ID** as folder name, and saves the notebook to a file named `NotebookName.ipynb`. For example, if you specify the Amazon S3 location `s3://MyBucket/MyNotebooks` for a notebook named `MyFirstEMRManagedNotebook`, the notebook file is saved to `s3://MyBucket/MyNotebooks/NotebookID/MyFirstEMRManagedNotebook.ipynb`.

8. Optionally, choose **Tags**, and then add any additional key-value tags for the notebook.

Important

A default tag with the **Key** string set to `creatorUserID` and the value set to your IAM user ID is applied for access purposes. We recommend that you do not change or remove this tag because it can be used to control access. For more information, see [Using Tags to Control User Permissions \(p. 31\)](#).

Creating Amazon EMR Clusters for Notebooks

When you create a notebook or change clusters, you can have Amazon EMR create a new cluster along with the notebook, or you can choose a cluster that was created earlier. Creating a new cluster with a notebook is a quick way to get started. Creating a cluster beforehand is useful if you need to install additional applications, connect to the cluster using SSH, or need other customizations that Amazon EMR offers when you create a cluster, such as user impersonation.

Notebook Limits Per Cluster

When you create a cluster that supports notebooks, consider the EC2 Instance type of the cluster master node. The memory constraints of this EC2 instance determine the number of notebooks that can be ready simultaneously to run code and queries on the cluster.

Master Node EC2 Instance Type	Number of Notebooks
*.medium	2
*.large	4
*.xlarge	8
*.2xlarge	16
*.4xlarge	24
*.8xlarge	24
*.16xlarge	24

Creating a Cluster When You Create a Notebook

When you have Amazon EMR create a cluster when you create an EMR notebook, the cluster has the following characteristics and limitations:

- It uses the most recent Amazon EMR release version and the versions of Hadoop, Spark, and Livy included with that release version. For more information, see the [Amazon EMR Release Guide](#).
- It is created without an EC2 key pair, so you are unable to connect to cluster EC2 instances using SSH. If an SSH connection is required, create a cluster first and then specify it when you create an EMR notebook.
- It uses On-Demand Instances, and the same instance type for all instances. One instance is used for the master node, and the remainder are used for core nodes.
- It uses the uniform instance groups configuration. For more information, see [Create a Cluster with Instance Fleets or Uniform Instance Groups](#) in the *Amazon EMR Management Guide*.
- It is launched in the default VPC for the AWS account.

You can specify a custom AWS service role and security groups, if required. For more information, see [Specifying the AWS Service Role \(p. 37\)](#) and [Specifying EC2 Security Groups for EMR Notebooks and Clusters \(p. 36\)](#). If you need additional customization or different settings, create a cluster beforehand using Amazon EMR, and then specify that cluster when you create the notebook.

Using an Existing Amazon EMR Cluster

EMR Notebooks supports clusters created only by using Amazon EMR. You can create a cluster using Amazon EMR if you require more processing power, storage, or any of the extensive cluster customization features that Amazon EMR offers. For more information about creating clusters, see [Plan and Configure Clusters](#) in the *Amazon EMR Management Guide*.

A cluster must meet the following requirements to be used with EMR Notebooks:

- The cluster must be launched within an EC2-VPC. Public and private subnets are supported. The EC2-Classic platform is not supported.

- The cluster must be created using Amazon EMR release version 5.18.0 or later.
- The cluster must be launched with Hadoop, Spark, and Livy installed. Other applications may be installed, but EMR Notebooks currently supports Spark clusters only.
- Clusters using Kerberos authentication are not supported.

Working with Notebooks

After you create an EMR notebook, the notebook takes a short time to start. The **Status** in the **Notebooks** list shows **Starting**. You can open a notebook when its status is **Ready**. It might take a bit longer for a notebook to be **Ready** if you created a cluster along with it.

Tip

Refresh your browser or choose the refresh icon above the notebooks list to refresh notebook status.

Understanding Notebook Status

An EMR notebook can have the following for **Status** in the **Notebooks** list.

Status	Meaning
Ready	You can open the notebook using the notebook editor. While a notebook has a Ready status, you can stop or delete it. To change clusters, you must stop the notebook first. If a notebook in the Ready status is idle for a long period of time, it is stopped automatically.
Starting	The notebook is being created and attached to the cluster. While a notebook is starting, you cannot open the notebook editor, stop it, delete it, or change clusters.
Pending	The notebook has been created, and is waiting for integration with the cluster to complete. The cluster may still be provisioning resources or responding to other requests. You can open the notebook editor with the notebook in <i>local mode</i> . Any code that relies on cluster processes does not execute and fails.
Stopping	The notebook is shutting down, or the cluster that the notebook is attached to is terminating. While a notebook is stopping, you can't open the notebook editor, stop it, delete it, or change clusters.
Stopped	The notebook has shut down. You can start the notebook on the same cluster, as long as the cluster is still running. You can change clusters, and delete the cluster.
Deleting	The cluster is being removed from the list of available clusters. The notebook file, <code>NotebookName.ipynb</code> remains in Amazon

Status	Meaning
	S3 and continues to accrue applicable storage charges.

Working with the Notebook Editor

An advantage of using an EMR notebook is that you can launch the Jupyter notebook editor directly from the console.

With EMR Notebooks, the notebook editor you access from the Amazon EMR console is the familiar open-source Jupyter Notebook editor. Because the notebook editor is launched within the Amazon EMR console, it's more efficient to configure access than it is with a notebook hosted on an Amazon EMR cluster. You don't need to configure a user's client to have web access through SSH, security group rules, and proxy configurations. If a user has sufficient permissions, they simply open the notebook editor within the Amazon EMR console.

Only one user can have an EMR notebook open at a time. If another user tries to open an EMR notebook that is already open, an error occurs.

To open the notebook editor for an EMR notebook

1. Select a notebook with a **Status** of **Ready** or **Pending** from the **Notebooks** list.
2. Choose **Open**.

A new browser tab opens to the Jupyter Notebook editor.

3. From the **Kernel** menu, choose **Change kernel** and then select the kernel for your programming language.

You are now ready to write and run code from within the notebook editor.

Saving the Contents of a Notebook

When you work in the notebook editor, the contents of notebook cells and output are saved automatically to the notebook file periodically in Amazon S3. A notebook that has no changes since the last time a cell was edited shows **(autosaved)** next to the notebook name in the editor. If changes have not yet been saved, **unsaved changes** appears.

You can save a notebook manually. From the **File** menu, choose **Save and Checkpoint** or press **CTRL+S**. This creates a file named **NotebookName.ipynb** in a **checkpoints** folder within the notebook folder in Amazon S3. For example, **s3://MyBucket/MyNotebookFolder/NotebookID/checkpoints/NotebookName.ipynb**. Only the most recent checkpoint file is saved in this location.

Changing Clusters

You can change the cluster that an EMR notebook is attached to without changing the contents of the notebook itself. You can change clusters for only those notebooks that have a **Stopped** status.

To change the cluster of an EMR notebook

1. If the notebook that you want to change is running, select it from the **Notebooks** list and choose **Stop**.
2. When the notebook status is **Stopped**, select the notebook from the **Notebooks** list, and then choose **View details**.

3. Choose **Change cluster**.
4. If you have an active cluster running Hadoop, Spark, and Livy to which you want to attach the notebook, leave the default, and select a cluster from the list. Only clusters that meet the requirements are listed.

—or—

Choose **Create a cluster** and then choose the cluster options. For more information, see [Creating a Cluster When You Create a Notebook \(p. 24\)](#).

5. Choose an option for **Security groups**, and then choose **Change cluster and start notebook**.

Deleting Notebooks and Notebook Files

When you delete an EMR notebook using the Amazon EMR console, you delete the notebook from the list of available notebooks. However, notebook files remain in Amazon S3 and continue to accrue storage charges.

To delete a notebook and remove associated files

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Notebooks**, select your notebook from the list, and then choose **View details**.
3. Choose the folder icon next to **Notebook location** and copy the **URL**, which is in the pattern `s3://MyNotebookLocationPath/NotebookID/`.
4. Choose **Delete**.

The notebook is removed from the list, and notebook details can no longer be viewed.

5. Follow the instructions for [How do I Delete Folders from an S3 Bucket](#) in the Amazon Simple Storage Service Console User Guide. Navigate to the bucket and folder from step 3.

—or—

If you have the AWS CLI installed, open a command prompt and type the command at the end of this paragraph. Replace the Amazon S3 location with the location that you copied above. Make sure that the AWS CLI is configured with the access keys of a user with permissions to delete the Amazon S3 location. For more information, see [Configuring the AWS CLI](#) in the *AWS Command Line Interface User Guide*.

```
aws s3 rm s3://MyNotebookLocationPath/NotebookID
```

Sharing Notebook Files

Each EMR notebook is saved to Amazon S3 as a file named `NotebookName.ipynb`. As long as a notebook file is compatible with the same version of Jupyter Notebook that EMR Notebooks is based on, you can open the notebook as an EMR notebook. You can replace the file for an EMR notebook with a different notebook file of the same name.

You can use this process to use EMR notebooks shared by others, notebooks shared in the Jupyter community, or to restore a notebook that was deleted from the console when you still have the notebook file.

To use a different notebook file as the basis for an EMR notebook

1. Before proceeding, close the notebook editor for any notebooks that you will work with, and then stop the notebook if it's an EMR notebook.

2. Create an EMR notebook and enter a name for it. The name that you enter for the notebook will be the name of the file you need to replace. The new file name must match this file name exactly.
3. Make a note of the location in Amazon S3 that you choose for the notebook. The file that you replace is in a folder with a path and file name like the following pattern:
`s3://MyNotebookLocation/NotebookID/MyNotebookName.ipynb`.
4. Stop the notebook.
5. Replace the old notebook file in the Amazon S3 location with the new one, using exactly the same name.

The following AWS CLI command for Amazon S3 replaces a file saved to a local machine called `SharedNotebook.ipynb` for an EMR notebook with the name **MyNotebook**, an ID of `e-12A3BCDEFJHIJKLMNOP45PQRST`, and created with `MyBucket/MyNotebooksFolder` specified in Amazon S3. For information about using the Amazon S3 console to copy and replace files, see [Uploading, Downloading, and Managing Objects in the Amazon Simple Storage Service Console User Guide](#).

```
aws s3 cp SharedNotebook.ipynb s3://MyBucket/  
MyNotebooksFolder/-12A3BCDEFJHIJKLMNOP45PQRST/MyNotebook.ipynb
```

Monitoring Spark User and Job Activity

EMR Notebooks allows you to configure user impersonation on a Spark cluster. This feature helps you track job activity initiated from within the notebook editor. In addition, EMR Notebooks has a built-in Jupyter Notebook widget that lets you view Spark job details alongside query output in the notebook editor. The widget is available by default and requires no special configuration. However, to view the history servers, your client must be configured to view Amazon EMR web interfaces that are hosted on the master node.

Setting Up Spark User Impersonation

By default, Spark jobs that users submit using the notebook editor appear to originate from an indistinct livy user identity. When you create a cluster to be used with EMR Notebooks, you can configure user impersonation for the cluster so that these jobs are associated with the IAM user identity that ran the code instead. HDFS user directories on the master node are created for each user identity that runs code in the notebook. For example, if user `NbUser1` runs code from the notebook editor, you can connect to the master node and see that `hadoop fs -ls /user` shows the directory `/user/user_NbUser1`.

You enable this feature by setting properties in the `core-site` and `livy-conf` configuration classifications when you create a cluster. This feature is not available by default when you have Amazon EMR create a cluster along with a notebook. For more information about using configuration classifications to customize applications, see [Configuring Applications](#) in the *Amazon EMR Release Guide*.

Use the following configuration classifications and values to enable user impersonation for EMR Notebooks:

```
[  
  {  
    "Classification": "core-site",  
    "Properties": {  
      "hadoop.proxyuser.livy.groups": "*",  
      "hadoop.proxyuser.livy.hosts": "*"  
    }  
  },  
  {  
    "Classification": "livy-conf",  
    "Properties": {  
    }  
  }]
```

```

        "livy.impersonation.enabled": "true"
    }
]

```

Using the Spark Job Monitoring Widget

When you run code in the notebook editor that execute Spark jobs on the EMR cluster, the output includes a Jupyter Notebook widget for Spark job monitoring. The widget provides job details and useful links to the Spark history server page and the Hadoop job history page, along with convenient links to job logs in Amazon S3 for any failed jobs.

To view history server pages on the cluster master node, you must set up an SSH client and proxy as appropriate. For more information, see [View Web Interfaces Hosted on Amazon EMR Clusters \(p. 243\)](#). To view logs in Amazon S3, cluster logging must be enabled, which is the default for new clusters. For more information, see [View Log Files Archived to Amazon S3 \(p. 213\)](#).

The following is an example of the Spark job monitoring widget.

The screenshot shows the Spark Job Progress widget with three sections:

- Job [0]: reduce at <stdin>:16**: Both Stage [0] and Stage [1] are COMPLETE. A callout box points to the section header with the text "Click to expand and view Spark job details".
- Job [1]: foreach at <stdin>:24**: Stage [2] is SKIPPED and Stage [3] is FAILED. A callout box points to the failed stage with the text "For failed jobs, click these links to view logs in Amazon S3 when logging is enabled on the cluster." Below the table, there are links for "stderr" and "stdout".
- Starting Spark application**: Shows a table of YARN Application IDs. One entry for application_1542497924776_0001 has a "Link" button under Driver log and another under Current session?. Callouts point to these buttons with the text "Click this link to view Spark History Server." and "Click this link to view Hadoop Job History." respectively.

EMR Notebooks Security and Access Control

Several features are available to help you tailor the security posture of EMR Notebooks. This helps ensure that only authorized users have access to an EMR notebook, can work with notebooks, and use the

notebook editor to execute code on the cluster. These features work along with the security features available for Amazon EMR and Amazon EMR clusters. For more information, see [Security](#) in the *Amazon EMR Management Guide*.

- You can use AWS Identity and Access Management policy statements together with notebook tags to limit access. For more information, see [Using Tags to Control User Permissions \(p. 31\)](#).
- EC2 security groups act as virtual firewalls that control network traffic between the cluster's master instance and the notebook editor. You can use defaults or customize these security groups. For more information, see [Specifying EC2 Security Groups for EMR Notebooks and Clusters \(p. 36\)](#).
- You specify an AWS Service Role that determines what permissions an EMR notebook has when interacting with other AWS services. For more information, see [Specifying the AWS Service Role \(p. 37\)](#).

IAM Policy Actions for EMR Notebooks

The actions listed in the following table are related to EMR Notebooks. For a user to perform an action, an IAM policy must be associated with the principal (user) that allows the action. For tagging purposes, some actions are performed on clusters, notebooks, or both. This is important for resource tagging. For more information, see [Using Tags to Control User Permissions \(p. 31\)](#).

Action	Description	Condition Key - Resource
<code>elasticmapreduce:CreateEditor</code>	Create an EMR notebook.	RequestTag - Notebooks ResourceTag - Clusters
<code>elasticmapreduce:StartEditor</code>	Start an EMR notebook.	ResourceTag - Notebooks, Clusters
<code>elasticmapreduce:StopEditor</code>	Stop an EMR notebook.	ResourceTag - Notebooks
<code>elasticmapreduce:OpenEditor</code>	Open the notebook editor from within the console.	ResourceTag - Notebooks, Clusters
<code>elasticmapreduce:DescribeEditor</code>	Editor an EMR notebook and view notebook properties.	ResourceTag - Notebooks
<code>elasticmapreduce>ListEditor</code>	Show the list of editors within the console.	ResourceTag - Notebooks
<code>elasticmapreduce>DeleteEditor</code>	Delete an EMR notebook using the console.	ResourceTag - Notebooks
	Warning This does not affect a user's ability to delete notebook files from Amazon S3. Use IAM policies for Amazon S3. For more information, see Using Bucket Policies and User Policies in the <i>Amazon Simple Storage Service Developer Guide</i> .	

Example –Minimal Policy Statement to Allow Full Access to EMR Notebooks

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Action": [  
                "elasticmapreduce:AddJobFlowSteps",  
                "iam:PassRole",  
                "elasticmapreduce:DescribeEditor",  
                "elasticmapreduce>ListEditors",  
                "elasticmapreduce:StartEditor",  
                "elasticmapreduce:StopEditor",  
                "elasticmapreduce:DeleteEditor",  
                "elasticmapreduce:OpenEditorInConsole"  
            ],  
            "Effect": "Allow",  
            "Resource": "*"  
        }  
    ]  
}
```

Using Tags to Control User Permissions

Permission for Amazon EMR actions associated with EMR Notebooks can be fine-tuned using tag-based access control in IAM policies. You can use a `Condition` element (also called a `Condition block`) to allow certain actions only when a notebook, cluster, or both has a certain tag key or key-value combination. You can also limit the `CreateEditor` action so that a request for a tag must be submitted when a user creates a notebook.

When you create an EMR notebook, a default tag is applied with a key string of `creatorUserId` set to the value of the IAM User ID who created the notebook. This is useful for limiting allowed actions for the notebook only to the creator.

The context keys that are available for Amazon EMR resources are available for notebooks as well:

- Use the `elasticmapreduce:ResourceTag`/`TagKeyString` condition context key to allow or deny user actions on clusters with tags that have the `TagKeyString` that you specify.
- Use the `elasticmapreduce:RequestTag`/`TagKeyString` condition context key along with the `CreateEditor` action to require that a key with `TagKeyString` is applied to a notebook when it's created.

For more information about using resource tags with actions that pass `ClusterID` as a required request parameter, see [Use Cluster Tagging with IAM Policies for Cluster-Specific Control \(p. 140\)](#).

If an action passes both `ClusterID` and `NotebookID` and you use the `ResourceTag` condition context key, the condition applies to both the cluster and the notebook. This means that both resources must have the tag key string or key-value combination. You can use the `Resource` element to limit the statement so that it applies only to clusters or notebooks as required. For more information, see the examples in the following sections.

Example Policy Statements Using Tags as Context Keys

The example IAM policy statements in this section demonstrate common scenarios for using keys to limit allowed actions using EMR Notebooks. As long as no other policy associated with the principal (user) allows the actions, the condition context keys limit allowed actions as indicated.

Example –Allow access only to notebooks that a user creates based on tagging

The example policy statement below, when attached to a role or user, allows the IAM user to work only with notebooks that they have created. This policy statement uses the default tag applied when a notebook is created.

In the example, the `StringEquals` condition operator tries to match a variable representing the current users IAM user ID (`{aws:userId}`) with the value of the tag `creatorUserId`. If the tag `creatorUserId` hasn't been added to the notebook, or doesn't contain the value of the current user's ID, 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 notebooks that have this tag with this value.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "elasticmapreduce:DescribeEditor",
                "elasticmapreduce:StartEditor",
                "elasticmapreduce:StopEditor",
                "elasticmapreduce:DeleteEditor",
                "elasticmapreduce:OpenEditorInConsole"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "elasticmapreduce:ResourceTag/creatorUserId": "{aws:userId}"
                }
            }
        }
    ]
}
```

Example –Require notebook tagging when a notebook is created

In this example, the `RequestTag` context key is used. The `CreateEditor` action is allowed only if the user does not change or delete the `creatorUserId` tag is added by default. The variable `#{aws:userId}`, specifies the currently active user's User ID, which is the default value of the tag.

The policy statement can be used to help ensure that users do not remove the `createUserId` tag or change its value.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "elasticmapreduce:CreateEditor"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "elasticmapreduce:RequestTag/creatorUserId": " #{aws:userid}"
                }
            }
        }
    ]
}
```

This example requires that the user create the cluster with a tag having the key string `dept` and a value set to one of the following: `datascience`, `analytics`, `operations`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "elasticmapreduce:CreateEditor"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "elasticmapreduce:RequestTag/dept": [
                        "datascience",
                        "analytics",
                        "operations"
                    ]
                }
            }
        }
    ]
}
```

Example –Limit notebook creation to tagged clusters, and require notebook tags

This example allows notebook creation only if the notebook is created with a tag that has the key string `owner` set to one of the specified values. In addition, the notebook can be created only if the cluster has a tag with the key string `department` set to one of the specified values.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "elasticmapreduce:CreateEditor"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "elasticmapreduce:RequestTag/owner": [
                        "owner1",
                        "owner2",
                        "owner3"
                    ],
                    "elasticmapreduce:ResourceTag/department": [
                        "dep1",
                        "dep3"
                    ]
                }
            }
        }
    ]
}
```

Example –Limit the ability to start a notebook based on tags

This example limits the ability to start notebooks only to those notebooks that have a tag with the key string `owner` set to one of the specified values. Because the `Resource` element is used to specify only the editor, the condition does not apply to the cluster, and it does not need to be tagged.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "elasticmapreduce:StartEditor"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:elasticmapreduce:*:123456789012:editor/*",
            "Condition": {
                "StringEquals": {
                    "elasticmapreduce:ResourceTag/owner": [
                        "owner1",
                        "owner2"
                    ]
                }
            }
        }
    ]
}
```

This example is similar to one above. However, the limit only applies to tagged clusters, not notebooks.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "elasticmapreduce:StartEditor"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:elasticmapreduce:*:123456789012:cluster/*",
            "Condition": {
                "StringEquals": {
                    "elasticmapreduce:ResourceTag/department": [
                        "dep1",
                        "dep3"
                    ]
                }
            }
        }
    ]
}
```

This example uses a different set of notebook and cluster tags. It allows a notebook to be started only if:

- The notebook has a tag with the key string `owner` set to any of the specified values
—and—
- The cluster has a tag with the key string `department` set to any of the specified values

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "elasticmapreduce:StartEditor"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:elasticmapreduce:*:123456789012:editor/*",
            "Condition": {

```

```

        "StringEquals": {
            "elasticmapreduce:ResourceTag/owner": [
                "user1",
                "user2"
            ]
        }
    },
    {
        "Action": [
            "elasticmapreduce:StartEditor"
        ],
        "Effect": "Allow",
        "Resource": "arn:aws:elasticmapreduce:*:123456789012:cluster/*",
        "Condition": {
            "StringEquals": {
                "elasticmapreduce:ResourceTag/department": [
                    "datascience",
                    "analytics"
                ]
            }
        }
    }
]
}

```

Example –Limit the ability to open the notebook editor based on tags

This example allows the notebook editor to be opened only if:

- The notebook has a tag with the key string `owner` set to any of the specified values.
—and—
- The cluster has a tag with the key string `department` set to any of the specified values.

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "elasticmapreduce:OpenEditorInConsole"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:elasticmapreduce:*:123456789012:editor/*",
            "Condition": {
                "StringEquals": {
                    "elasticmapreduce:ResourceTag/owner": [
                        "user1",
                        "user2"
                    ]
                }
            }
        },
        {
            "Action": [
                "elasticmapreduce:OpenEditorInConsole"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:elasticmapreduce:*:123456789012:cluster/*",
            "Condition": {
                "StringEquals": {
                    "elasticmapreduce:ResourceTag/department": [
                        "datascience",

```

```
        "analytics"
    ]
}
}
```

Specifying EC2 Security Groups for EMR Notebooks and Clusters

When you create an EMR notebook, two security groups are used to control network traffic between the EMR notebook and the Amazon EMR cluster when the notebook editor is used. The default security groups have minimal rules that allow only network traffic between the EMR Notebooks service and the clusters to which notebooks are attached.

An EMR notebook uses [Apache Livy](#) to communicate with the cluster via a proxy using TCP Port 18888. By creating custom security groups with rules tailored to your environment, you can limit network traffic so that only a subset of notebooks can run code within the notebook editor on particular clusters. The security groups are used in addition to the security groups for the cluster. For more information, see [Control Network Traffic with Security Groups](#) in the *Amazon EMR Management Guide* and [Specifying EC2 Security Groups for EMR Notebooks and Clusters \(p. 36\)](#).

Default EC2 Security Group for the Master Instance

The default EC2 security group for the master instance is associated with the master instance in addition to the cluster's security groups for the master instance.

Group Name: **ElasticMapReduceEditors-Livy**

Rules

- Inbound

Allow TCP Port 18888 from any resources in the default EC2 security group for EMR Notebooks

- Outbound

None

EC2 Security Group for EMR Notebooks

The default EC2 security group for the EMR Notebooks is associated with the notebook editor for any EMR notebook to which it is assigned.

Group Name: **ElasticMapReduceEditors-Editor**

Rules

- Inbound

None

- Outbound

Allow TCP Port 18888 to any resources in the default EC2 security group for EMR Notebooks.

Specifying the AWS Service Role

Each EMR notebook needs permissions to access other AWS resources and perform actions. The notebook's *AWS service role* defines these permissions. The IAM policies attached to the service role provide permissions for the cluster to interoperate with other AWS services.

The default service role for an EMR notebook is **EMR_Notebooks_DefaultRole**. The default permissions policy attached to this role are in the default managed policy, **EMRNNotebooksDefaultRolePolicy**. The contents of this policy are listed as follows:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "ec2:AuthorizeSecurityGroupEgress",  
                "ec2:AuthorizeSecurityGroupIngress",  
                "ec2>CreateSecurityGroup",  
                "ec2:DescribeSecurityGroups",  
                "ec2:RevokeSecurityGroupEgress",  
                "ec2>CreateNetworkInterface",  
                "ec2:CreateNetworkInterfacePermission",  
                "ec2>DeleteNetworkInterface",  
                "ec2:DeleteNetworkInterfacePermission",  
                "ec2:DescribeNetworkInterfaces",  
                "ec2:ModifyNetworkInterfaceAttribute",  
                "ec2:DescribeTags",  
                "ec2:DescribeInstances",  
                "ec2:DescribeSubnets",  
                "elasticmapreduce>ListInstances",  
                "elasticmapreduce>DescribeCluster"  
            ],  
            "Resource": "*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": "ec2>CreateTags",  
            "Resource": "arn:aws:ec2:*::network-interface/*",  
            "Condition": {  
                "ForAllValues:StringEquals": {  
                    "aws:TagKeys": [  
                        "aws:elasticmapreduce:editor-id",  
                        "aws:elasticmapreduce:job-flow-id"  
                    ]  
                }  
            }  
        }  
    ]  
}
```

The default role also has the **S3FullAccessPolicy** attached. The contents of this policy are as follows:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "s3:*",  
            "Resource": "*"  
        }  
    ]
```

}

Plan and Configure Clusters

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 \(p. 39\)](#).
- Whether a cluster is long-running or transient, and what software it runs. See [Configuring a Cluster to Auto-Terminate or Continue \(p. 79\)](#) and [Configure Cluster Software \(p. 91\)](#).
- The hardware and networking options that optimize cost, performance, and availability for your application. See [Configure Cluster Hardware and Networking \(p. 96\)](#).
- How to set up clusters so you can manage them more easily, and monitor activity, performance, and health. See [Configure Cluster Logging and Debugging \(p. 126\)](#) and [Tag Clusters \(p. 130\)](#).
- How to authenticate and authorize access to cluster resources, and how to encrypt data. See [Security \(p. 136\)](#).
- How to integrate with other software and services. See [Drivers and Third-Party Application Integration \(p. 134\)](#).

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 \(p. 39\)](#)
- [Work with Storage and File Systems \(p. 40\)](#)
- [Prepare Input Data \(p. 43\)](#)
- [Configure an Output Location \(p. 51\)](#)

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, see 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 the Hadoop file system 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>
(Legacy) Amazon S3 block file system	<code>s3bfs://</code>	<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`.

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
```

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 \(p. 43\)](#)
- [How to Get Data Into Amazon EMR \(p. 43\)](#)

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, which 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, see <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 (Hive 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-premises data to process, you may find the AWS Direct Connect service useful.

Topics

- [Upload Data to Amazon S3 \(p. 43\)](#)
- [Import files with Distributed Cache \(p. 47\)](#)
- [How to Process Compressed Files \(p. 50\)](#)
- [Import DynamoDB Data into Hive \(p. 50\)](#)
- [Connect to Data with AWS DirectConnect \(p. 50\)](#)
- [Upload Large Amounts of Data with AWS Import/Export \(p. 50\)](#)

Upload Data to Amazon S3

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

Topics

- [Create and Configure an Amazon S3 Bucket \(p. 44\)](#)
- [Configure Multipart Upload for Amazon S3 \(p. 44\)](#)
- [Best Practices \(p. 46\)](#)

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, see [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. You can also create and set permissions for an Amazon S3 bucket using the Amazon S3 API or AWS CLI. You can also use Curl along with a modification to pass the appropriate authentication parameters for Amazon S3.

See the following resources:

- To create a bucket using the console, see [Create a Bucket in the Amazon Simple Storage Service Console User Guide](#).
- To create and work with buckets using the AWS CLI, see [Using High-Level S3 Commands with the AWS Command Line Interface in the Amazon Simple Storage Service Console User Guide](#).
- To create a bucket using an SDK, see [Examples of Creating a Bucket in the Amazon Simple Storage Service Developer Guide](#).
- To work with buckets using Curl, see [Amazon S3 Authentication Tool for Curl](#).
- For more information on specifying Region-specific buckets, see [Accessing a Bucket in the Amazon Simple Storage Service Developer Guide](#).

Note

If you enable logging for a bucket, it enables only bucket access logs, not Amazon EMR cluster logs.

During bucket creation or after, you can set the appropriate permissions to access the bucket depending on your application. Typically, you give yourself (the owner) read and write access and give authenticated users read access.

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.

Configure Multipart Upload for Amazon S3

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, see [Multipart Upload Overview in the Amazon Simple Storage Service Developer Guide](#).

In addition, Amazon EMR offers properties that allow you to more precisely control the clean up of failed multipart upload parts.

The following table describes the Amazon EMR configuration properties for multipart upload. You can configure these using the core-site configuration classification when you create a cluster. For more information, see [Configure Applications in the Amazon EMR Release Guide](#).

Configuration Parameter Name	Default Value	Description
<code>fs.s3n.multipart.uploads.enabled</code>		A boolean type that indicates whether to enable multipart uploads. When EMRFS Consistent View (p. 56) is enabled, multipart uploads are enabled by default and setting this value to <code>false</code> is ignored.
<code>fs.s3n.multipart.uploads.splitSize8</code>		Specifies the maximum size of a part, in bytes, before EMRFS starts a new part upload when multipart uploads is enabled. The minimum value is 5242880 (5 MB). If a lesser value is specified, 5242880 is used. The maximum is 5368709120 (5 GB). If a greater value is specified, 5368709120 is used. If EMRFS client-side encryption is disabled and the Amazon S3 Optimized Committer is also disabled, this value also controls the maximum size that a data file can grow until EMRFS uses multipart uploads rather than a <code>PutObject</code> request to upload the file. For more information, see
<code>fs.s3n.ssl.enabled</code>	<code>true</code>	A boolean type that indicates whether to use <code>http</code> or <code>https</code> .
<code>fs.s3.buckets.create.enabled</code>	<code>true</code>	A boolean type that indicates whether a bucket should be created if it does not exist. Setting to <code>false</code> causes an exception on <code>CreateBucket</code> operations.
<code>fs.s3.multipart.clean.enabled</code>	<code>false</code>	A boolean type that indicates whether to enable background periodic clean up of incomplete multipart uploads.
<code>fs.s3.multipart.clean.age.threshold</code>		A long type that specifies the minimum age of a multipart upload, in seconds, before it is considered for cleanup. The default is one week.
<code>fs.s3.multipart.clean.jitter</code>	<code>10m</code>	An integer type that specifies the maximum amount of random jitter delay in seconds added to the 15-minute fixed delay before scheduling next round of clean up.

Disable Multipart Upload Using the Amazon EMR Console

This procedure explains how to disable multipart upload using the Amazon EMR console when you create a cluster.

To disable multipart uploads

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**, **Go to advanced options**.
3. Under **Edit Software Settings** and enter the following configuration: `classification=core-site, properties=[fs.s3.multipart.uploads.enabled=false]`

4. Proceed with creating the cluster.

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-actions` parameter.

To disable multipart upload using the AWS CLI

1. Create a file, `myConfig.json`, with the following contents and save it in the same directory where you run the command:

```
[  
  {  
    "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.

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-5.20.0 --applications Name=Hive Name=Pig \  
--use-default-roles --ec2-attributes KeyName=myKey --instance-type m4.large \  
--instance-count 3 --configurations file://myConfig.json
```

Disable Multipart Upload Using the API

For information on using Amazon S3 multipart uploads programmatically, see [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, see [AWS SDK for Java](#).

Best Practices

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, see [Using Versioning](#) in the *Amazon Simple Storage Service Developer Guide*.

Clean Up Failed Multipart Uploads

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 by default. For information about changing properties related to this configuration using Amazon EMR, see [Configure Multipart Upload for Amazon S3 \(p. 44\)](#). 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. We recommend the following options to avoid excessive file storage:

- For buckets that you use with Amazon EMR, use a lifecycle configuration rule in Amazon S3 to remove incomplete multipart uploads three days after the upload initiation date. Lifecycle configuration rules allow you to control the storage class and lifetime of objects. For more information, see [Object Lifecycle Management](#), and [Aborting Incomplete Multipart Uploads Using a Bucket Lifecycle Policy](#).
- Enable Amazon EMR's multipart cleanup feature by setting `fs.s3.multipart.clean.enabled` to `TRUE` and tuning other cleanup parameters. This feature is useful at high volume, large scale, and with clusters that have limited uptime. In this case, the `DaysAfterInitiation` parameter of a lifecycle configuration rule may be too long, even if set to its minimum, causing spikes in Amazon S3 storage. Amazon EMR's multipart cleanup allows more precise control. For more information, see [Configure Multipart Upload for Amazon S3 \(p. 44\)](#).

Manage Version Markers

We recommend that you enable a lifecycle configuration rule in Amazon S3 to remove expired object delete markers for versioned buckets that you use with Amazon EMR. 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 in the bucket. While you are not charged for delete markers, removing expired markers can improve the performance of LIST requests. 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 a bucket. For more information, see [Request Rate and Performance Considerations](#) in the *Amazon Simple Storage Service Developer Guide*.

Import files with Distributed Cache

Topics

- [Supported File Types \(p. 47\)](#)
- [Location of Cached Files \(p. 48\)](#)
- [Access Cached Files From Streaming Applications \(p. 48\)](#)
- [Access Cached Files From Streaming Applications Using the Amazon EMR Console \(p. 48\)](#)
- [Access Cached Files From Streaming Applications Using the AWS CLI \(p. 49\)](#)

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 [Create Bootstrap Actions to Install Additional Software \(p. 92\)](#).

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 core 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 core nodes only. If there are no core 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	Specify <code>-cacheFile</code> followed by the name	<code>-cacheFile \</code>	

If you want to ...	Action	Example	
file to the Distributed Cache	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.	s3://bucket_name/file_name#cache_file_name	
Add an archive file to the Distributed Cache	Enter -cacheArchive 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.	-cacheArchive \ s3://bucket_name/ archive_name#cache_archive_name	

6. 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 -cacheFile 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 -cacheArchive 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 <https://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 m4.large --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_file" ]
```

```
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 m4.large --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 \(p. 51\)](#)
- [What formats can Amazon EMR return? \(p. 52\)](#)
- [How to write data to an Amazon S3 bucket you don't own \(p. 52\)](#)
- [Compress the Output of your Cluster \(p. 54\)](#)

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 \(p. 39\)](#) 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 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/MapperScript.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
<code>AuthenticatedRead</code>	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.
<code>BucketOwnerFullControl</code>	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.
<code>BucketOwnerRead</code>	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.
<code>LogDeliveryWrite</code>	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.
<code>Private</code>	Specifies that the owner is granted <code>Permission.FullControl</code> .
<code>PublicRead</code>	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.
<code>PublicReadWrite</code>	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 \(p. 239\)](#).

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;
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 \(p. 239\)](#).

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 \(p. 55\)](#)

- [Using the Snappy Library with Amazon EMR \(p. 55\)](#)

Output Data Compression

This compresses the output of your Hadoop job. If you are using TextOutputFormat the result is a gzip'ed 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-actions 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. Compress the map output and decompress it when it arrives on the core 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/>.

Use EMR File System (EMRFS)

The EMR File System (EMRFS) is an implementation of HDFS that all Amazon EMR clusters use 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 consistent view and data encryption.

Consistent view provides consistency checking for list and read-after-write (for new put requests) for objects in Amazon S3. Data encryption allows you to encrypt objects that EMRFS writes to Amazon S3, and enables EMRFS to work with encrypted objects in Amazon S3. If you are using Amazon EMR release version 4.8.0 or later, you can use security configurations to set up encryption for EMRFS objects in Amazon S3, along with other encryption settings. For more information, see [Encryption Options \(p. 157\)](#). If you use an earlier release version of Amazon EMR, you can manually configure encryption settings. For more information, see [Specifying Amazon S3 Encryption Using EMRFS Properties \(p. 71\)](#).

When using Amazon EMR release version 5.10.0 or later, you can use different IAM roles for EMRFS requests to Amazon S3 based on cluster users, groups, or the location of EMRFS data in Amazon S3. For more information, see [Configure IAM Roles for EMRFS Requests to Amazon S3 \(p. 164\)](#).

Topics

- [Consistent View \(p. 56\)](#)
- [Authorizing Access to EMRFS Data in Amazon S3 \(p. 70\)](#)
- [Specifying Amazon S3 Encryption Using EMRFS Properties \(p. 71\)](#)

Consistent View

EMRFS consistent view is an optional feature available when using Amazon EMR release version 3.2.1 or later. Consistent view allows EMR clusters to check for list and read-after-write consistency for Amazon S3 objects written by or synced with EMRFS. Consistent view addresses an issue that can arise due to the [Amazon S3 Data Consistency Model](#). For example, if you add objects to Amazon S3 in one operation and then immediately list objects in a subsequent operation, the list and the set of objects processed may be incomplete. This is more commonly a problem for clusters that run quick, sequential steps using Amazon S3 as a data store, such as multi-step extract-transform-load (ETL) data processing pipelines.

When you create a cluster with consistent view enabled, Amazon EMR uses an Amazon DynamoDB database to store object metadata and track consistency with Amazon S3. If consistent view determines that Amazon S3 is inconsistent during a file system operation, it retries that operation according to rules that you can define. By default, the DynamoDB database has 500 read capacity and 100 write capacity. You can configure read/write capacity settings depending on the number of objects that EMRFS tracks and the number of nodes concurrently using the metadata. You can also configure other database and operational parameters. Using consistent view incurs DynamoDB charges, which are typically small, in addition to the charges for Amazon EMR. For more information, see [Amazon DynamoDB Pricing](#).

With consistent view enabled, EMRFS returns the set of objects listed in an EMRFS metadata store and those returned directly by Amazon S3 for a given path. 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 that you track are checked for consistency.

You can use the EMRFS utility (`emrfs`) from the command line of the master node to perform operations on Amazon S3 objects that are tracked by consistent view. For example, you can import, delete, and sync Amazon S3 objects with the EMRFS metadata store. For more information about the EMRFS CLI utility, see [EMRFS CLI Reference \(p. 64\)](#).

If you directly delete objects from Amazon S3 that are tracked in EMRFS metadata, EMRFS treats the object as inconsistent and throws an exception after it has exhausted retries. Use EMRFS to delete objects in Amazon S3 that are tracked using consistent view. Alternatively, you can use the `emrfs` command line to purge metadata entries for objects that have been directly deleted, or you can sync the consistent view with Amazon S3 immediately after you delete the objects.

Topics

- [Enable Consistent View \(p. 57\)](#)
- [Understanding How EMRFS Consistent View Tracks Objects in Amazon S3 \(p. 58\)](#)
- [Retry Logic \(p. 58\)](#)
- [EMRFS Consistent View Metadata \(p. 59\)](#)
- [Configure Consistency Notifications for CloudWatch and Amazon SQS \(p. 61\)](#)
- [Configure Consistent View \(p. 62\)](#)
- [EMRFS CLI Reference \(p. 64\)](#)

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. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**, **Go to advanced options**.
3. Choose settings for **Step 1: Software and Steps** and **Step 2: Hardware**.
4. For **Step 3: General Cluster Settings**, under **Additional Options**, choose **EMRFS consistent view**.
5. For **EMRFS Metadata store**, type the name of your metadata store. The default value is `EmrFSSMetadata`. If the EmrFSSMetadata 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.

6. For **Number of retries**, type an integer value. If an inconsistency is detected, EMRFS tries to call Amazon S3 this number of times. The default value is **5**.
7. For **Retry period (in seconds)**, type an integer value. This is the amount of time that EMRFS waits between retry attempts. 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 that 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 m4.large --instance-count 3 --emrfs
  Consistent=true \
  --release-label emr-5.20.0 --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.

Understanding How EMRFS Consistent View Tracks Objects in Amazon S3

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 EMRFS deletes an object, 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 [EMRFS CLI Reference \(p. 64\)](#). For more information about purging listings in the EMRFS metadata, see [EMRFS Consistent View Metadata \(p. 59\)](#).

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 retries the operation according to parameters defined in `emrfs-site` configuration properties. After EMRFS exhausts the retries, it either throws a `ConsistencyException` or logs the exception and continue the workflow. For more information about retry logic, see [Retry Logic \(p. 58\)](#). You can find `ConsistencyExceptions` in your logs, for example:

- `listStatus`: No Amazon S3 object for metadata item `/S3_bucket/dir/object`
- `getFileStatus`: Key `dir/file` is present in metadata but not Amazon S3

If you delete an object directly from Amazon S3 that EMRFS consistent view tracks, EMRFS treats that object as inconsistent because it is still listed in the metadata as present in Amazon S3. If your metadata becomes out of sync with the objects EMRFS tracks in Amazon S3, you can use the `sync` sub-command of the EMRFS CLI to reset metadata so that it reflects Amazon S3. To discover discrepancies between metadata and Amazon S3, use the `diff`. 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 returns the superset of the objects being tracked in the metadata and those in that Amazon S3 path.

Retry Logic

EMRFS tries 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 creates a cluster with consistency enabled, which logs inconsistencies and sets a fixed retry interval of 10 seconds:

Example Setting retry period to a fixed amount

```
aws emr create-cluster --release-label emr-5.20.0 \
--instance-type m4.large --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 \(p. 56\)](#).

EMRFS Consistent View Metadata

EMRFS consistent view tracks consistency using a DynamoDB table to track objects in Amazon S3 that 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. Multiple clusters can share the same metadata.

How to add entries to metadata

You can use the `sync` or `import` subcommands to add entries to metadata. `sync` reflects 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 [EMRFS CLI Reference \(p. 64\)](#).

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 [EMRFS CLI Reference \(p. 64\)](#).

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 are 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 attempts 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 Amazon 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 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 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 differs 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
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 using configuration properties for `emrfs-site` properties. For example, you can choose a different default DynamoDB throughput by supplying the following arguments to the CLI `--emrfs` option, using the `emrfs-site` configuration classification (Amazon EMR release version 4.x and later only), or a bootstrap action to configure the `emrfs-site.xml` file on the master node:

Example Changing default metadata read and write values at cluster launch

```
aws emr create-cluster --release-label emr-5.20.0 --instance-type m4.large \  
--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-5.20.0 --applications Name=Hive \  
--instance-type m4.large --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 CLI Command Reference](#).

`emrfs-site.xml` properties for consistent view

Property	Default value	Description
<code>fs.s3.consistent</code>	<code>false</code>	When set to <code>true</code> , this property configures EMRFS to use DynamoDB to provide consistency.

Property	Default value	Description
<code>fs.s3.consistent.retryPolicyType</code>	<code>exponential</code>	This property identifies the policy to use when retrying for consistency issues. Options include: exponential, fixed, or none.
<code>fs.s3.consistent.retryPeriodSeconds</code>	<code>10</code>	This property sets the length of time to wait between consistency retry attempts.
<code>fs.s3.consistent.retryCount</code>	<code>5</code>	This property sets the maximum number of retries when inconsistency is detected.
<code>fs.s3.consistent.throwExceptionOnInconsistency</code>		This property determines whether to throw or log a consistency exception. When set to <code>true</code> , a <code>ConsistencyException</code> is thrown.
<code>fs.s3.consistent.metadata.autoCreate</code>	<code>true</code>	When set to <code>true</code> , this property enables automatic creation of metadata tables.
<code>fs.s3.consistent.metadata.tableName</code>	<code>EmrFSMetadata</code>	This property specifies the name of the metadata table in DynamoDB.
<code>fs.s3.consistent.metadata.read.capacity</code>	<code>400</code>	This property specifies the DynamoDB read capacity to provision when the metadata table is created.
<code>fs.s3.consistent.metadata.write.capacity</code>	<code>100</code>	This property specifies the DynamoDB write capacity to provision when the metadata table is created.
<code>fs.s3.consistent.fastList</code>	<code>true</code>	When set to <code>true</code> , 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>	<code>false</code>	When set to <code>true</code> , this property enables metadata prefetching for directories containing more than 20,000 items.
<code>fs.s3.consistent.notification.CloudWatch</code>	<code>false</code>	When set to <code>true</code> , CloudWatch metrics are enabled for FileSystem API calls that fail due to Amazon S3 eventual consistency issues.
<code>fs.s3.consistent.notification.SQS</code>	<code>false</code>	When set to <code>true</code> , eventual consistency notifications are pushed to an Amazon SQS queue.

Property	Default value	Description
<code>fs.s3.consistent.notification.SQS.queue</code>	<code>EMRFS-Inconsistency- <jobFlowId></code>	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.customMessage</code>		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>	<code>none</code>	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 Amazon EMR release version 3.2.1 or later. You can use the EMRFS CLI to manage the metadata for 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 AWS_ACCESS_KEY_ID</code> <code>--access-key AWS_ACCESS_KEY_ID</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, <code>AWS_ACCESS_KEY_ID</code> is set to the access key used to create the cluster.	No
<code>-s AWS_SECRET_ACCESS_KEY</code> <code>--secret-key AWS_SECRET_ACCESS_KEY</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, <code>AWS_SECRET_ACCESS_KEY</code> 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

Option	Description	Required
<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 METADATA_NAME --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

Example 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 METADATA_NAME --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>-r READ_CAPACITY --read-capacity READ_CAPACITY</code>	The requested read throughput capacity for the metadata table. If the <code>READ_CAPACITY</code> argument is not supplied, the default value is 400.	No
<code>-w WRITE_CAPACITY --write-capacity WRITE_CAPACITY</code>	The requested write throughput capacity for the metadata table. If the <code>WRITE_CAPACITY</code> argument is not supplied, the default value is 100.	No

Example 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 EmrFSMetadata.	No

Example 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 EmrFSMetadata.	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

Example 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].

Example 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
<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 EmrFSMetadata.	No
<code>s3://<i>s3Path</i></code>	The path to the Amazon S3 bucket to compare with the metadata table. Buckets sync recursively.	Yes

Example 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 EmrFSMetadata.	No
<code>s3://<i>s3Path</i></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

Option	Description	Required
--write-consumption <i>WRITE_CONSUMPTION</i>	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

Example 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
-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 EmrFSMetadata.	No
<i>s3://s3Path</i>	The path to the Amazon S3 bucket you are tracking for consistent view. Buckets sync recursively.	Yes
--read-consumption <i>READ_CONSUMPTION</i>	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
--write-consumption <i>WRITE_CONSUMPTION</i>	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

Example 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
-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 EmrFSMetadata.	No

Option	Description	Required
<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

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

emrfs read-sqs sub-command

[options] for emrfs read-sqs

Option	Description	Required
<code>-q QUEUE_NAME --queue-name QUEUE_NAME</code>	<code>QUEUE_NAME</code> 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 OUTPUT_FILE --output-file OUTPUT_FILE</code>	<code>OUTPUT_FILE</code> 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 QUEUE_NAME --queue-name QUEUE_NAME</code>	<code>QUEUE_NAME</code> 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.

Authorizing Access to EMRFS Data in Amazon S3

By default, the EMR role for EC2 determines the permissions for accessing EMRFS data in Amazon S3. The IAM policies that are attached to this role apply regardless of the user or group making the request through EMRFS. The default is `EMR_EC2_DefaultRole`. For more information, see [Use Default IAM Roles and Managed Policies \(p. 188\)](#).

Beginning with Amazon EMR release version 5.10.0, you can use a security configuration to specify IAM roles for EMRFS. This allows you to customize permissions for EMRFS requests to Amazon S3 for clusters that have multiple users. You can specify different IAM roles for different users and groups, and for different Amazon S3 bucket locations based on the prefix in Amazon S3. When EMRFS makes a request to Amazon S3 that matches users, groups, or the locations that you specify, the cluster uses the corresponding role that you specify instead of the EMR role for EC2. For more information, see [Configure IAM Roles for EMRFS Requests to Amazon S3 \(p. 164\)](#).

Alternatively, if your Amazon EMR solution has demands beyond what IAM roles for EMRFS provides, you can define a custom credentials provider class, which allows you to customize access to EMRFS data in Amazon S3.

Creating a Custom Credentials Provider for EMRFS Data in Amazon S3

To create a custom credentials provider, you implement the `AWSCredentialsProvider` and the `Hadoop Configurable` classes.

For a detailed explanation of this approach, see [Securely Analyze Data from Another AWS Account with EMRFS](#) in the AWS Big Data blog. The blog post includes a tutorial that walks you through the process end-to-end, from creating IAM roles to launching the cluster. It also provides a Java code example that implements the custom credential provider class.

The basic steps are as follows:

To specify a custom credentials provider

1. Create a custom credentials provider class compiled as a JAR file.
2. Run a script as a bootstrap action to copy the custom credentials provider JAR file to the `/usr/share/aws/emr/emrfs/auxlib` location on the cluster's master node. For more information about bootstrap actions, see [\(Optional\) Create Bootstrap Actions to Install Additional Software](#).
3. Customize the `emrfs-site` classification to specify the class that you implement in the JAR file. For more information about specifying configuration objects to customize applications, see [Configuring Applications](#) in the *Amazon EMR Release Guide*.

The following example demonstrates a `create-cluster` command that launches a Hive cluster with common configuration parameters, and also includes:

- A bootstrap action that runs the script, `copy_jar_file.sh`, which is saved to `mybucket` in Amazon S3.
- An `emrfs-site` classification that specifies a custom credentials provider defined in the JAR file as `MyCustomCredentialsProvider`

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 --applications Name=Hive \
--bootstrap-actions '[{"Path":"s3://mybucket/copy_jar_file.sh","Name":"Custom
action"}]' \
--ec2-attributes '{"KeyName":"MyKeyPair","InstanceProfile":"EMR_EC2_DefaultRole",\
"SubnetId":"subnet-xxxxxxxx","EmrManagedSlaveSecurityGroup":"sg-xxxxxxxx",\
"EmrManagedMasterSecurityGroup":"sg-xxxxxxxx"}' \
--service-role EMR_DefaultRole --enable-debugging --release-label emr-5.20.0 \
--log-uri 's3n://my-emr-log-bucket/' --name 'test-awscredentialsprovider-emrfs' \
--instance-type=m4.large --instance-count 3 \
--configurations '[{"Classification":"emrfs-site",\
"Properties": {"fs.s3.customAWSCredentialsProvider": "MyAWSCredentialsProviderWithUri"},\
"Configurations": []}]'
```

Specifying Amazon S3 Encryption Using EMRFS Properties

Important

Beginning with Amazon EMR release version 4.8.0, you can use security configurations to apply encryption settings more easily and with more options. We recommend using security configurations. For information, see [Configure Data Encryption \(p. 176\)](#). The console instructions described in this section are available for release versions earlier than 4.8.0. If you use the AWS CLI to configure Amazon S3 encryption both in the cluster configuration and in a security configuration in subsequent versions, the security configuration overrides the cluster configuration.

When you create a cluster, you can specify server-side encryption (SSE) or client-side encryption (CSE) for EMRFS data in Amazon S3 using the console or using `emrfs-site` classification properties through the AWS CLI or EMR SDK. Amazon S3 SSE and CSE are mutually exclusive; you can choose either but not both.

For AWS CLI instructions, see the appropriate section for your encryption type below.

To specify EMRFS encryption options using the AWS Management Console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**, **Go to advanced options**.
3. Choose a **Release** of 4.7.2 or earlier.
4. Choose other options for **Software and Steps** as appropriate for your application, and then choose **Next**.
5. Choose settings in the **Hardware and General Cluster Settings** panes as appropriate for your application.
6. On the **Security** pane, under **Authentication and encryption**, select the **S3 Encryption (with EMRFS)** option to use.

Note

S3 server-side encryption with KMS Key Management (SSE-KMS) is not available when using Amazon EMR release version 4.4 or earlier.

- If you choose an option that uses **AWS Key Management**, choose an **AWS KMS Key ID**. For more information, see [Using AWS KMS Customer Master Keys \(CMKs\) for EMRFS Encryption \(p. 72\)](#).
 - If you choose **S3 client-side encryption with custom materials provider**, provide the **Class name** and the **JAR location**. For more information, see [Amazon S3 Client-Side Encryption \(p. 74\)](#).
7. Choose other options as appropriate for your application and then choose **Create Cluster**.

Using AWS KMS Customer Master Keys (CMKs) for EMRFS Encryption

The AWS KMS encryption key must be created in the same region as your Amazon EMR cluster instance and the Amazon S3 buckets used with EMRFS. If the key that you specify is in a different account from the one that you use to configure a cluster, you must specify the key using its ARN.

The role for the Amazon EC2 instance profile must have permissions to use the CMK you specify. The default role for the instance profile in Amazon EMR is `EMR_EC2_DefaultRole`. If you use a different role for the instance profile, or you use IAM roles for EMRFS requests to Amazon S3, make sure that each role is added as a key user as appropriate. This gives the role permissions to use the CMK. For more information, see [Using Key Policies in the AWS Key Management Service Developer Guide](#) and [Use Default IAM Roles and Managed Policies \(p. 188\)](#). Although you may use the same AWS KMS customer master key (CMK) for Amazon S3 data encryption as you use for local disk encryption, using separate keys is recommended.

You can use the AWS Management Console to add your instance profile or EC2 instance profile to the list of key users for the specified AWS KMS CMK, or you can use the AWS CLI or an AWS SDK to attach an appropriate key policy.

The procedure below describes how to add the default EMR instance profile, `EMR_EC2_DefaultRole` as a *key user* using the AWS Management Console. It assumes that you have already created a CMK. To create a new CMK, see [Creating Keys in the AWS Key Management Service Developer Guide](#).

To add the EC2 instance profile for Amazon EMR to the list of encryption key users

1. Sign in to the AWS Management Console and open the AWS Key Management Service (AWS KMS) console at <https://console.aws.amazon.com/kms>.
2. To change the AWS Region, use the Region selector in the upper-right corner of the page.
3. Select the alias of the CMK to modify.
4. On the key details page under **Key Users**, choose **Add**.
5. In the **Attach** dialog box, select the appropriate role. The name of the default role is `EMR_EC2_DefaultRole`.

6. Choose **Attach**.

Amazon S3 Server-Side Encryption

When you set up Amazon S3 server-side encryption, Amazon S3 encrypts data at the object level as it writes the data to disk and decrypts the data when it is accessed. For more information about SSE, see [Protecting Data Using Server-Side Encryption](#) in the *Amazon Simple Storage Service Developer Guide*.

You can choose between two different key management systems when you specify SSE in Amazon EMR:

- **SSE-S3:** Amazon S3 manages keys for you.
- **SSE-KMS:** You use an AWS KMS customer master key (CMK) set up with policies suitable for Amazon EMR. For more information about key requirements for Amazon EMR, see [Using AWS KMS Customer Master Keys \(CMKs\) for Encryption \(p. 161\)](#). When you use AWS KMS, charges apply for the storage and use of encryption keys. For more information, see [AWS KMS Pricing](#).

SSE with customer-provided keys (SSE-C) is not available for use with Amazon EMR.

To create a cluster with SSE-S3 enabled using the AWS CLI

- Type the following command:

```
aws emr create-cluster --release-label emr-4.7.2 or earlier \
--instance-count 3 --instance-type m4.large --emrfs Encryption=ServerSide
```

You can also enable SSE-S3 by setting the `fs.s3.enableServerSideEncryption` property to true in `emrfs-site` properties. See the example for SSE-KMS below and omit the property for Key ID.

To create a cluster with SSE-KMS enabled using the AWS CLI

Note

SSE-KMS is available only in Amazon EMR release version 4.5.0 and later.

- Type the following AWS CLI command to create a cluster with SSE-KMS, where `keyID` is an AWS KMS customer master key (CMK), for example, `a4567b8-9900-12ab-1234-123a45678901`:

```
aws emr create-cluster --release-label emr-4.7.2 or earlier --instance-count 3 \
--instance-type m4.large --use-default-roles \
--emrfs Encryption=ServerSide,Args=[fs.s3.serverSideEncryption.kms.keyId=keyID]
```

--OR--

Type the following AWS CLI command using the `emrfs-site` classification and provide a configuration JSON file with contents as shown similar to `myConfig.json` in the example below:

```
aws emr create-cluster --release-label emr-4.7.2 or earlier --instance-count 3 --
instance-type m4.large --applications Name=Hadoop --configurations file://myConfig.json
--use-default-roles
```

Example contents of `myConfig.json`:

```
[{"Classification": "emrfs-site", "Properties": {}}
```

```

        "fs.s3.enableServerSideEncryption": "true",
        "fs.s3.serverSideEncryption.kms.keyId": "a4567b8-9900-12ab-1234-123a45678901"
    }
]
```

Configuration Properties for SSE-S3 and SSE-KMS

These properties can be configured using the `emrfs-site` configuration classification. SSE-KMS is available only in Amazon EMR release version 4.5.0 and later.

Property	Default value	Description
<code>fs.s3.enableServerSideEncryption</code>	<code>false</code>	When set to <code>true</code> , objects stored in Amazon S3 are encrypted using server-side encryption. If no key is specified, SSE-S3 is used.
<code>fs.s3.serverSideEncryption.kms.keyId</code>	<code>n/a</code>	Specifies an AWS KMS key ID or ARN. If a key is specified, SSE-KMS is used.

Amazon S3 Client-Side Encryption

With Amazon S3 client-side encryption, the Amazon S3 encryption and decryption takes place in the EMRFS client on your cluster. Objects are encrypted before being uploaded to Amazon S3 and decrypted after they are downloaded. The provider you specify supplies the encryption key that the client uses. The client can use keys provided by AWS KMS (CSE-KMS) or a custom Java class that provides the client-side master key (CSE-C). The encryption specifics are slightly different between CSE-KMS and CSE-C, depending on the specified provider and the metadata of the object being decrypted or encrypted. For more information about these differences, see [Protecting Data Using Client-Side Encryption](#) in the [Amazon Simple Storage Service Developer Guide](#).

Note

Amazon S3 CSE only ensures that EMRFS data exchanged with Amazon S3 is encrypted; not all data on cluster instance volumes is encrypted. Furthermore, because Hue does not use EMRFS, objects that the Hue S3 File Browser writes to Amazon S3 are not encrypted.

To specify CSE-KMS for EMRFS data in Amazon S3 using the AWS CLI

- Type the following command and replace `MyKMSKeyId` with the Key ID or ARN of the AWS KMS CMK to use:

```
aws emr create-cluster --release-label emr-4.7.2 or earlier
--emrfs Encryption=ClientSide,ProviderType=KMS,KMSKeyId=MyKMSKeyId
```

Creating a Custom Key Provider

When you create a custom key provider, the application is expected to implement the [EncryptionMaterialsProvider interface](#), which is available in the AWS SDK for Java version 1.11.0 and later. The implementation can use any strategy to provide encryption materials. You may, for example, choose to provide static encryption materials or integrate with a more complex key management system.

The encryption algorithm used for custom encryption materials must be **AES/GCM/NoPadding**.

The `EncryptionMaterialsProvider` class gets encryption materials by encryption context. Amazon EMR populates encryption context information at runtime to help the caller determine the correct encryption materials to return.

Example Example: Using a Custom Key Provider for Amazon S3 Encryption with EMRFS

When Amazon EMR fetches the encryption materials from the `EncryptionMaterialsProvider` class to perform encryption, EMRFS optionally populates the `materialsDescription` argument with two fields: the Amazon S3 URI for the object and the `JobFlowId` of the cluster, which can be used by the `EncryptionMaterialsProvider` class to return encryption materials selectively.

For example, the provider may return different keys for different Amazon S3 URI prefixes. It is the description of the returned encryption materials that is eventually stored with the Amazon S3 object rather than the `materialsDescription` value that is generated by EMRFS and passed to the provider. While decrypting an Amazon S3 object, the encryption materials description is passed to the `EncryptionMaterialsProvider` class, so that it can, again, selectively return the matching key to decrypt the object.

An `EncryptionMaterialsProvider` reference implementation is provided below. Another custom provider, [EMRFSRSAEncryptionMaterialsProvider](#), is available from GitHub.

```
import com.amazonaws.services.s3.model.EncryptionMaterials;
import com.amazonaws.services.s3.model.EncryptionMaterialsProvider;
import com.amazonaws.services.s3.model.KMSEncryptionMaterials;
import org.apache.hadoop.conf.Configurable;
import org.apache.hadoop.conf.Configuration;

import java.util.Map;

/**
 * Provides KMSEncryptionMaterials according to Configuration
 */
public class MyEncryptionMaterialsProviders implements EncryptionMaterialsProvider,
Configurable{
    private Configuration conf;
    private String kmsKeyId;
    private EncryptionMaterials encryptionMaterials;

    private void init() {
        this.kmsKeyId = conf.get("my.kms.key.id");
        this.encryptionMaterials = new KMSEncryptionMaterials(kmsKeyId);
    }

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

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

    @Override
    public void refresh() {

    }

    @Override
    public EncryptionMaterials getEncryptionMaterials(Map<String, String>
materialsDescription) {
        return this.encryptionMaterials;
```

```
    }

    @Override
    public EncryptionMaterials getEncryptionMaterials() {
        return this.encryptionMaterials;
    }
}
```

Specifying a Custom Materials Provider Using the AWS CLI

To use the AWS CLI, pass the `Encryption`, `ProviderType`, `CustomProviderClass`, and `CustomProviderLocation` arguments to the `emrfs` option.

```
aws emr create-cluster --instance-type m4.large --release-label emr-4.7.2 or earlier --
emrfs Encryption=ClientSide,ProviderType=Custom,CustomProviderLocation=s3://mybucket/
myfolder/provider.jar,CustomProviderClass=classname
```

Setting `Encryption` to `ClientSide` enables client-side encryption, `CustomProviderClass` is the name of your `EncryptionMaterialsProvider` object, and `CustomProviderLocation` is the local or Amazon S3 location from which Amazon EMR copies `CustomProviderClass` to each node in the cluster and places it in the classpath.

Specifying a Custom Materials Provider Using an SDK

To use an SDK, you can set the property `fs.s3.cse.encryptionMaterialsProvider.uri` to download the custom `EncryptionMaterialsProvider` class that you store in Amazon S3 to each node in your cluster. You configure this in `emrfs-site.xml` file along with CSE enabled and the proper location of the custom provider.

For example, in the AWS SDK for Java using `RunJobFlowRequest`, your code might look like the following:

```
<snip>
    Map<String, String> emrfsProperties = new HashMap<String, String>();
    emrfsProperties.put("fs.s3.cse.encryptionMaterialsProvider.uri", "s3://mybucket/
MyCustomEncryptionMaterialsProvider.jar");
    emrfsProperties.put("fs.s3.cse.enabled", "true");
    emrfsProperties.put("fs.s3.consistent", "true");

    emrfsProperties.put("fs.s3.cse.encryptionMaterialsProvider", "full.class.name.of.EncryptionMaterialsPro
Configuration myEmrfsConfig = new Configuration()
    .withClassification("emrfs-site")
    .withProperties(emrfsProperties);

RunJobFlowRequest request = new RunJobFlowRequest()
    .withName("Custom EncryptionMaterialsProvider")
    .withReleaseLabel("emr-5.20.0")
    .withApplications(myApp)
    .withConfigurations(myEmrfsConfig)
    .withServiceRole("EMR_DefaultRole")
    .withJobFlowRole("EMR_EC2_DefaultRole")
    .withLogUri("s3://myLogUri/")
    .withInstances(new JobFlowInstancesConfig()
        .withEc2KeyName("myEc2Key")
        .withInstanceCount(2)
        .withKeepJobFlowAliveWhenNoSteps(true)
        .withMasterInstanceType("m4.large")
        .withSlaveInstanceType("m4.large")
    );

```

```
RunJobFlowResult result = emr.runJobFlow(request);
</snip>
```

Custom EncryptionMaterialsProvider with Arguments

You may need to pass arguments directly to the provider. To do this, you can use the `emrfs-site` configuration classification with custom arguments defined as properties. An example configuration is shown below, which is saved as a file, `myConfig.json`:

```
[
  {
    "Classification": "emrfs-site",
    "Properties": {
      "myProvider.arg1": "value1",
      "myProvider.arg2": "value2"
    }
  }
]
```

Using the `create-cluster` command from the AWS CLI, you can use the `--configurations` option to specify the file as shown below:

```
aws emr create-cluster --release-label emr-5.20.0 --instance-type m4.large
--instance-count 2 --configurations file://myConfig.json --emrfs
Encryption=ClientSide,CustomProviderLocation=s3://mybucket/myfolder/
myprovider.jar,CustomProviderClass=classname
```

`emrfs-site.xml` Properties for Amazon S3 Client-Side Encryption

Property	Default value	Description
<code>fs.s3.cse.enabled</code>	<code>false</code>	When set to <code>true</code> , EMRFS objects stored in Amazon S3 are encrypted using client-side encryption.
<code>fs.s3.cse.encryptionMaterialsProvider.uri</code>	<code>N/A</code>	Applies when using custom encryption materials. The Amazon S3 URI where the JAR with the <code>EncryptionMaterialsProvider</code> is located. When you provide this URI, Amazon EMR automatically downloads the JAR to all nodes in the cluster.
<code>fs.s3.cse.encryptionMaterialsProvider</code>	<code>N/A</code>	The <code>EncryptionMaterialsProvider</code> class path used with client-side encryption. When using CSE-KMS, specify <code>com.amazon.ws.emr.hadoop.fs.cse.KMSEncryptionMaterialsProvider</code> .
<code>fs.s3.cse.materialsDescription.enabled</code>	<code>false</code>	When set to <code>true</code> , populates the <code>materialsDescription</code> of encrypted objects with the Amazon S3 URI for the object and the <code>JobFlowId</code> .

Property	Default value	Description
		Set to <code>true</code> when using custom encryption materials.
<code>fs.s3.cse.kms.keyId</code>	<code>N/A</code>	Applies when using CSE-KMS. The value of the KeyId, ARN, or alias of the AWS KMS CMK used for encryption.
<code>fs.s3.cse.cryptoStorageMode</code>	<code>ObjectMetadata</code>	The Amazon S3 storage mode. By default, the description of the encryption information is stored in the object metadata. You can also store the description in an instruction file. Valid values are <code>ObjectMetadata</code> and <code>InstructionFile</code> . For more information, see Client-Side Data Encryption with the AWS SDK for Java and Amazon S3 .

Control Cluster Termination

When you create a cluster using Amazon EMR, you can choose to create a transient cluster that auto-terminates after steps complete, or you can create a long-running cluster that continues to run until you terminate it deliberately. When a cluster terminates, all Amazon EC2 instances in the cluster terminate, and data in the instance store and EBS volumes is no longer available and not recoverable. Understanding and managing cluster termination is critical to developing a strategy to manage and preserve data by writing to Amazon S3 and balancing cost. For information about how to terminate a cluster manually, see [Terminate a Cluster \(p. 251\)](#).

When you use auto-termination, the cluster starts, runs any bootstrap actions that you specify, and then executes steps that typically input data, process the data, and then produce and save output. When the steps finish, Amazon EMR automatically terminates the cluster Amazon EC2 instances. This is an effective model for a cluster that performs a periodic processing task, such as a daily data processing run. Auto-terminating a cluster helps ensure that you are billed only for the time required to process your data. For more information about steps, see [Work with Steps Using the CLI and Console \(p. 270\)](#).

With a long-running cluster, the cluster starts the same way. You can specify steps as you would with a cluster that terminates automatically, but the cluster continues to run and accrue charges after steps complete. This model is effective when you need to interactively or automatically query data, or interact with big data applications hosted on the cluster on an ongoing basis. It is also effective if you periodically process a data set so large or so frequently that it is inefficient to launch new clusters and load data each time. You can enable termination protection on long-running clusters to help prevent accidental shutdown. You can also take advantage of features like automatic scaling and instance fleets to dynamically size the cluster to balance performance and cost in response to workload demands. For more information, see [Scaling Cluster Resources \(p. 253\)](#) and [Configure Instance Fleets \(p. 112\)](#).

This section describes how termination protection and auto-termination work, and how they interact with one another, other Amazon EMR features, and other data processes.

Topics

- [Configuring a Cluster to Auto-Terminate or Continue \(p. 79\)](#)
- [Using Termination Protection \(p. 79\)](#)

Configuring a Cluster to Auto-Terminate or Continue

By default, clusters that you create using the console or the AWS CLI continue to run until you shut them down. To have a cluster terminate after running steps, you need to enable auto-termination. In contrast, clusters that you launch using the EMR API have auto-termination enabled by default.

To disable auto-termination using the EMR API

- When using the [RunJobFlow](#) action to create a cluster, set the [KeepJobFlowAliveWhenNoSteps](#) property to `true`.

To enable auto-termination using Quick Options in the AWS Management Console

- Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
- Choose **Create cluster**.
- Choose **Step execution**.
- Choose other settings as appropriate for your application, and then choose **Create cluster**.

To enable auto-termination using Advanced Options in the AWS Management Console

- Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
- Choose **Create cluster**.
- Choose **Go to advanced options**.
- Under **Add steps (optional)** select **Auto-terminate cluster after the last step is completed**.
- Choose other settings as appropriate for your application, and then choose **Create cluster**.

To enable auto-termination 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-5.20.0 \
--applications Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey \
--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/] \
--instance-type m4.large --instance-count 3 --auto-terminate
```

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

Using Termination Protection

When termination protection is enabled on a long-running cluster, you can still terminate the cluster, but you must explicitly remove termination protection from the cluster first. This helps ensure that EC2 instances are not shut down by an accident or error. Termination protection is especially useful if

your cluster might have data stored on local disks that you need to recover before the instances are terminated. You can enable termination protection when you create a cluster, and you can change the setting on a running cluster.

With termination protection enabled, the `TerminateJobFlows` action in the Amazon EMR API does not work. Users cannot terminate the cluster using this API or the `terminate-clusters` command from the AWS CLI. The API returns an error, and the CLI exits with a non-zero return code. When using the Amazon EMR console to terminate a cluster, you are prompted with an extra step to turn termination protection off.

Warning

Termination protection does not guarantee that data is retained in the event of a human error or a workaround—for example, if a reboot command is issued from the command line while connected to the instance using SSH, if an application or script running on the instance issues a reboot command, or if the Amazon EC2 or Amazon EMR API is used to disable termination protection. Even with termination protection enabled, data saved to instance storage, including HDFS data, can be lost. Write data output to Amazon S3 locations and create backup strategies as appropriate for your business continuity requirements.

Termination protection does not affect your ability to scale cluster resources using any of the following actions:

- Resizing a cluster manually using the AWS Management Console or AWS CLI. For more information, see [Manually Resizing a Running Cluster \(p. 262\)](#).
- Removing instances from a core or task instance group using a scale-in policy with automatic scaling. For more information, see [Using Automatic Scaling in Amazon EMR \(p. 254\)](#).
- Removing instances from an instance fleet by reducing target capacity. For more information, see [Instance Fleet Options \(p. 112\)](#).

Termination Protection and Amazon EC2

An Amazon EMR cluster with termination protection enabled has the `disableAPITermination` attribute set for all Amazon EC2 instances in the cluster. If a termination request originates with Amazon EMR, and the Amazon EMR and Amazon EC2 settings for an instance conflict, the Amazon EMR setting overrides the Amazon EC2 setting. For example, if you use the Amazon EC2 console to *enable* termination protection on an Amazon EC2 instance in a cluster that has termination protection *disabled*, when you use the Amazon EMR console, AWS CLI commands for Amazon EMR, or the Amazon EMR API to terminate the cluster, Amazon EMR sets `DisableApiTermination` to `false` and terminates the instance along with other instances.

Important

If an instance is created as part of an Amazon EMR cluster with termination protection, and the Amazon EC2 API or AWS CLI commands are used to modify the instance so that `DisableApiTermination` is `false`, and then the Amazon EC2 API or AWS CLI commands execute the `TerminateInstances` action, the Amazon EC2 instance terminates.

Termination Protection and Unhealthy YARN Nodes

Amazon EMR periodically checks the Apache Hadoop YARN status of nodes running on core and task Amazon EC2 instances in a cluster. The health status is reported by the [NodeManager Health Checker Service](#). If a node reports UNHEALTHY, the Amazon EMR instance controller blacklists the node and does not allocate YARN containers to it until it becomes healthy again. A common reason for unhealthy nodes is that disk utilization goes above 90%. For more information about identifying unhealthy nodes and recovering, see [Resource Errors \(p. 290\)](#).

If the node remains UNHEALTHY for more than 45 minutes, Amazon EMR takes the following action based on the status of termination protection.

Termination Protection	Result
Enabled (Recommended)	The Amazon EC2 instance remains in a blacklisted state and continues to count toward cluster capacity. You can connect to the Amazon EC2 instance for configuration and data recovery, and resize your cluster to add capacity. For more information, see Resource Errors (p. 290) .
Disabled	<p>The Amazon EC2 instance is terminated. Amazon EMR provisions a new instance based on the specified number of instances in the instance group or the target capacity for instance fleets. If all core nodes are UNHEALTHY for more than 45 minutes, the cluster terminates, reporting a NO_SLAVES_LEFT status.</p> <p>Important HDFS data may be lost if a core instance terminates because of an unhealthy state. If the node stored blocks that were not replicated to other nodes, these blocks are lost, which might lead to data loss. We recommend that you use termination protection so that you can connect to instances and recover data as necessary.</p>

Termination Protection, Auto-Termination, and Step Execution

The auto-terminate setting takes precedence over termination protection. If both are enabled, when steps finish executing, the cluster terminates instead of entering a waiting state.

When you submit steps to a cluster, you can set the `ActionOnFailure` property to determine what happens if the step can't complete execution because of an error. The possible values for this setting are `TERMINATE_CLUSTER` (`TERMINATE_JOB_FLOW` with earlier versions), `CANCEL_AND_WAIT`, and `CONTINUE`. For more information, see [Work with Steps Using the CLI and Console \(p. 270\)](#).

If a step fails that is configured with `ActionOnFailure` set to `CANCEL_AND_WAIT`, if auto-termination is enabled, the cluster terminates without executing subsequent steps.

If a step fails that is configured with `ActionOnFailure` set to `TERMINATE_CLUSTER`, use the table of settings below to determine the outcome.

ActionOnFailure	Auto-Termination	Termination Protection	Result
TERMINATE_CLUSTER	Enabled	Disabled	Cluster terminates
	Enabled	Enabled	Cluster terminates
	Disabled	Enabled	Cluster continues
	Disabled	Disabled	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 Spot price.

Configuring Termination Protection When You Launch a Cluster

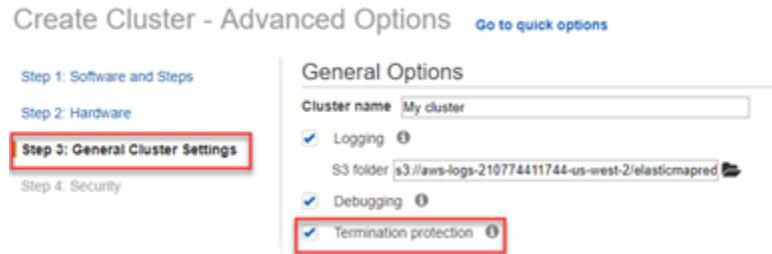
You can enable or disable termination protection when you launch a cluster using the console, the AWS CLI, or the API.

The default termination protection setting depends on how you launch the cluster:

- Amazon EMR Console Quick Options—Termination Protection is **disabled** by default.
- Amazon EMR Console Advanced Options—Termination Protection is **enabled** by default.
- AWS CLI `aws emr create-cluster`—Termination Protection is **disabled** unless `--termination-protected` is specified.
- Amazon EMR API `RunJobFlow` command—Termination Protection is **disabled** unless the `TerminationProtected` boolean value is set to `true`.

To enable or disable termination protection when creating a 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. For **Step 3: General Cluster Settings**, under **General Options** make sure **Termination protection** is selected to enable it, or clear the selection to disable it.



5. Choose other settings as appropriate for your application, choose **Next**, and then finish configuring your cluster.

To enable termination protection when creating a cluster using the AWS CLI

- Using the AWS CLI, you can launch a cluster with termination protection enabled by using the `create-cluster` command with the `--termination-protected` parameter. Termination protection is disabled by default.

The following example creates cluster with termination protection enabled:

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 "TerminationProtectedCluster" --release-label emr-5.20.0 \
\ --applications Name=Hadoop Name=Hive Name=Pig \
```

```
--use-default-roles --ec2-attributes KeyName=myKey --instance-type m4.large \
--instance-count 3 --termination-protected
```

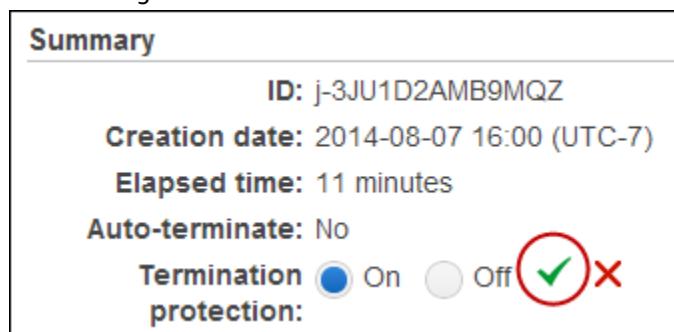
For more information about using Amazon EMR commands in the AWS CLI, see <https://docs.aws.amazon.com/cli/latest/reference/emr>.

Configuring Termination Protection for Running Clusters

You can configure termination protection for a running cluster using the console or the AWS CLI.

To enable or disable 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 **Clusters** page, choose the **Name** of your cluster.
3. On the **Summary** tab, for **Termination protection**, choose **Change**.
4. To enable termination protection, choose **On**. To disable termination protection, choose **Off**. Then choose the green check mark to confirm.



To enable or disable termination protection for a running cluster using the AWS CLI

- To enable termination protection on a running cluster using the AWS CLI, use the `modify-cluster-attributes` command with the `--termination-protected` parameter. To disable it, use the `--no-termination-protected` parameter.

The following example enables termination protection on the cluster with ID `j-3KVTXXXXXX7UG`:

```
aws emr modify-cluster-attributes --cluster-id j-3KVTXXXXXX7UG --termination-protected
```

The following example disables termination protection on the same cluster:

```
aws emr modify-cluster-attributes --cluster-id j-3KVTXXXXXX7UG --no-termination-protected
```

Working with Amazon Linux AMIs in Amazon EMR

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 the configurations required for each instance to host your cluster applications.

By default, when you create a cluster, Amazon EMR uses a default Amazon Linux AMI that is created specifically for the Amazon EMR release version you use. When you use Amazon EMR 5.7.0 or later, you

can choose to specify a custom Amazon Linux AMI instead of the default Amazon Linux AMI for Amazon EMR. A custom AMI allows you to encrypt the root device volume and to customize applications and configurations as an alternative to using bootstrap actions.

Amazon EMR automatically attaches an Amazon EBS General Purpose SSD volume as the root device for all AMIs. Using an EBS-backed AMI enhances performance. The EBS costs are pro-rated by the hour based on the monthly Amazon EBS charges for gp2 volumes in the region where the cluster runs. For example, the cost per hour for the root volume on each cluster instance in a region that charges \$0.10/GB/month is approximately \$0.00139 per hour (\$0.10/GB/month divided by 30 days divided by 24h times 10 GB). Whether you use the default Amazon Linux AMI or a custom Amazon Linux AMI, you can specify the size of the EBS root device volume from 10-100 GiB.

For more information about Amazon Linux AMIs, see [Amazon Machine Images \(AMI\)](#). For more information about instance storage for Amazon EMR instances, see [Instance Store and Amazon EBS \(p. 101\)](#).

Topics

- [Using the Default Amazon Linux AMI for Amazon EMR \(p. 84\)](#)
- [Using a Custom AMI \(p. 85\)](#)
- [Specifying the Amazon EBS Root Device Volume Size \(p. 90\)](#)

Using the Default Amazon Linux AMI for Amazon EMR

Each Amazon EMR release version uses a default Amazon Linux AMI for Amazon EMR unless you specify a custom AMI. The default AMI is based on the most up-to-date Amazon Linux AMI available at the time of the Amazon EMR release. The AMI is tested for compatibility with the big-data applications and Amazon EMR features included with that release version.

Each Amazon EMR release version is "locked" to the Amazon Linux AMI version to maintain compatibility. This means that the same Amazon Linux AMI version is used for an Amazon EMR release version even when newer Amazon Linux AMIs become available. For this reason, we recommend that you use the latest Amazon EMR release version (currently 5.20.0) unless you need an earlier version for compatibility and are unable to migrate.

If you must use an earlier release version of Amazon EMR for compatibility, we recommend that you use the latest release in a series. For example, if you must use the 5.12 series, use 5.12.2 instead of 5.12.0 or 5.12.1. If a new release becomes available in a series, consider migrating your applications to the new release.

How Software Updates Are Managed

When an Amazon EC2 instance in a cluster that is based on the default Amazon Linux AMI for Amazon EMR boots for the first time, it checks the enabled package repositories for Amazon Linux and Amazon EMR for software updates that apply to the AMI version. As with other Amazon EC2 instances, critical and important security updates from these repositories are installed automatically. For more information, see [Package Repository](#) in the *Amazon EC2 User Guide for Linux Instances*. Other software packages and kernel updates are not installed because that risks introducing compatibility errors.

When you connect to a cluster instance using SSH, the first few lines of screen output provide a link to the release notes for the Amazon Linux AMI that the instance uses, a notice of the most recent Amazon Linux AMI version, a notice of the number of packages available for update from the enabled repositories, and a directive to run `sudo yum update`.

Important

We strongly recommend that you do not run `sudo yum update` on cluster instances, either while connected using SSH or using a bootstrap action. This might cause incompatibilities because all packages are installed indiscriminately.

Best Practices for Managing Software Updates

- If you use an earlier release version of Amazon EMR, consider and test a migration to the latest release before updating software packages.
- If you migrate to a later release version or you upgrade software packages, test the implementation in a non-production environment first. The option to clone clusters using the Amazon EMR management console is helpful for this.
- Evaluate software updates for your applications and for your version of Amazon Linux AMI on an individual basis. Only test and install packages in production environments that you determine to be absolutely necessary for your security posture, application functionality, or performance.
- Watch the [Amazon Linux Security Center](#) for updates.
- Avoid installing packages by connecting to individual cluster instances using SSH. Instead, use a bootstrap action to install and update packages on all cluster instances as necessary. This requires that you terminate a cluster and relaunch it. For more information, see [Create Bootstrap Actions to Install Additional Software \(p. 92\)](#).

Using a Custom AMI

When you use Amazon EMR 5.7.0 or later, you can choose to specify a custom Amazon Linux AMI instead of the default Amazon Linux AMI for Amazon EMR. A custom AMI is useful if you want to do the following:

- 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 \(p. 88\)](#).

Note

You do not need a custom AMI to encrypt instance storage and EBS storage volumes. You can use an Amazon EMR security configuration to do this when using either the Amazon Linux AMI for Amazon EMR or a custom AMI. 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. This 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 \(p. 87\)](#).
- Implement more sophisticated cluster and node configurations than bootstrap actions allow.

Best Practices and Considerations

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

- You must use an Amazon Linux AMI. Amazon Linux 2 AMIs are not supported. Only 64-bit Amazon Linux AMIs are supported. Amazon Linux AMIs with multiple Amazon EBS volumes are not supported.
- Base your customization on the most recent EBS-backed [Amazon Linux AMI](#). For a list of Amazon Linux AMIs and corresponding AMI IDs, see [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 \(p. 98\)](#).

- 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.
- If you use a custom Amazon Linux AMI based on an Amazon Linux AMI with a creation date of 2018-08-11, the Oozie server fails to start. If you use Oozie, create a custom AMI based on an Amazon Linux AMI ID with a different creation date. You can use the following AWS CLI command to return a list of Image IDs for all HVM Amazon Linux AMIs with a 2018.03 version, along with the release date, so that you can choose an appropriate Amazon Linux AMI as your base. Replace MyRegion with your region identifier, such as us-west-2.

```
aws ec2 --region MyRegion describe-images --owner amazon --query 'Images[?Name!=`null`]&[?starts_with(Name, `amzn-ami-hvm-2018.03`)==`true`].[CreationDate,ImageId,Name]' --output text | sort -rk1
```

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, [Amazon CloudWatch](#), or the Amazon EMR API. The AMI must exist in the same AWS 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 \(p. 87\)](#).

The screenshot shows the 'Additional Options' section of the AWS Management Console. It includes fields for 'EMRFS consistent view' (unchecked), 'Custom AMI ID' (set to 'ami-*xxxxxxxx*', with a dropdown arrow and an info icon), and 'Update all installed packages on reboot (recommended)' (checked). Below this section is a link to '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 \(p. 90\)](#).

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 "Cluster with My Custom AMI" \
--custom-ami-id MyAmiID --ebs-root-volume-size 20 \
--release-label emr-5.7.0 --use-default-roles \
--instance-count 2 --instance-type m4.large
```

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 \(p. 88\)](#).

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 and `MyAmiID` with the ID of an appropriate Amazon Linux AMI. For the most recent AMI IDs, see [Amazon Linux AMI](#).

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 ec2 run-instances --image-id MyAmiID \
--count 1 --instance-type m4.large \
--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 \(p. 86\)](#) and [Creating a Custom AMI with an Encrypted Amazon EBS Root Device Volume \(p. 88\)](#).

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 that the following actions must be allowed:

- `kms:encrypt`
- `kms:decrypt`
- `kms:ReEncrypt*`
- `kms>CreateGrant`
- `kms:GenerateDataKeyWithoutPlaintext"`
- `kms:DescribeKey"`

The easiest way to do this is to add the role as a key user as described in the following tutorial. The following example policy statement is provided if you need to customize role policies.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "EmrDiskEncryptionPolicy",  
      "Effect": "Allow",  
      "Action": [  
        "kms:Encrypt",  
        "kms:Decrypt",  
        "kms:ReEncrypt*",  
        "kms>CreateGrant",  
        "kms:GenerateDataKeyWithoutPlaintext",  
        "kms:DescribeKey"  
      ],  
      "Resource": [  
        "*"  
      ]  
    }  
  ]  
}
```

Note

Local disk encryption using a security configuration requires that the Amazon EMR *instance profile* (usually `EMR_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. Sign in to the AWS Management Console and open the AWS Key Management Service (AWS KMS) console at <https://console.aws.amazon.com/kms>.
2. To change the AWS Region, use the Region selector in the upper-right corner of the page.
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.

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 ec2 copy-image --source-image-id MyAmiId \
--source-region us-west-2 --name MyEncryptedEMRAmi \
--encrypted --kms-key-id arn:aws:kms:us-west-2:12345678910:key/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx
```

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 \(p. 86\)](#). 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 \(p. 87\)](#).

Specifying the Amazon EBS Root Device Volume Size

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 \(p. 86\)](#).

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:

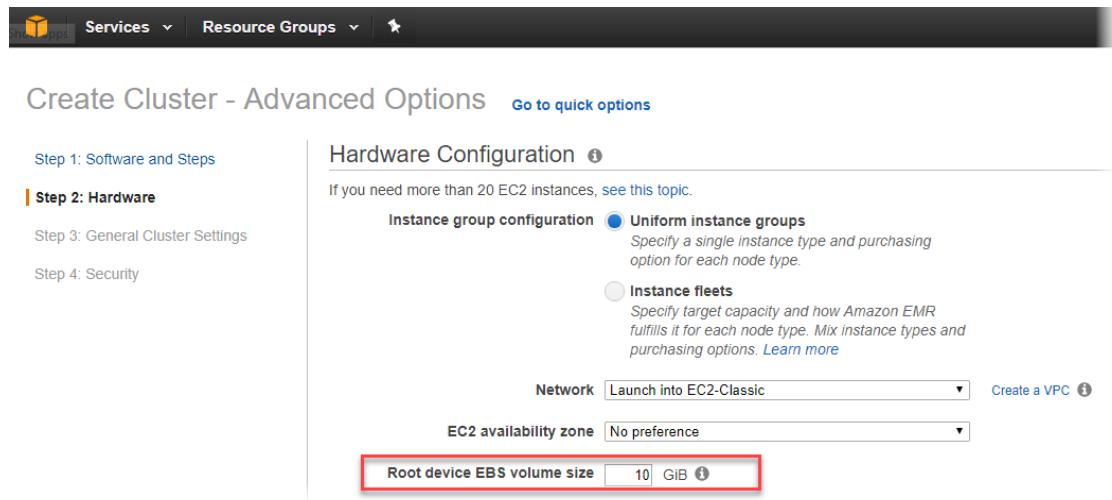
$$(\$/EBS\;GB/month) \times 0.931323 \div 30 \div 24 \times EMR_EBSRootGiB \times InstanceCount$$

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.



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

- Use the `--ebs-root-volume-size` parameter of the `create-cluster` command as shown in the following example.

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 --release-label emr-5.7.0 \
--ebs-root-volume-size 20 --instance-groups InstanceGroupType=MASTER, \
InstanceCount=1,InstanceType=m4.large \
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large
```

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.

Create Bootstrap Actions to Install Additional Software

You can use a *bootstrap action* to install additional software or customize the configuration of cluster instances. Bootstrap actions are scripts that run on cluster after Amazon EMR launches the instance using the Amazon Linux Amazon Machine Image (AMI). Bootstrap actions run before Amazon EMR installs the applications that you specify when you create the cluster and before cluster nodes begin processing data. If you add nodes to a running cluster, bootstrap actions also run on those nodes in the same way. You can create custom bootstrap actions and specify them when you create your cluster.

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 in Amazon EMR 4.x Release Versions](#) in the Amazon EMR Release Guide.

Topics

- [Bootstrap Action Basics \(p. 92\)](#)
- [Run If Bootstrap Action \(p. 93\)](#)
- [Shutdown Actions \(p. 93\)](#)
- [Use Custom Bootstrap Actions \(p. 94\)](#)

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-actions` 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-actions` parameter when you create the cluster using the `create-cluster` command. The syntax for a `--bootstrap-actions` parameter is as follows:

AWS CLI

```
--bootstrap-actions 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 m4.large --bootstrap-actions 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 <https://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 \(p. 239\)](#).

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 \(p. 94\)](#)
- [Add Custom Bootstrap Actions Using the Console \(p. 95\)](#)
- [Use a Custom Bootstrap Action to Copy an Object from Amazon S3 to Each Node \(p. 95\)](#)

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

The following example uses a bootstrap action script to download and extract 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 m4.large \
--bootstrap-actions 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 m4.large --bootstrap-actions 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 <https://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 \(p. 211\)](#)

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 a cluster before your applications are installed. The AWS CLI is installed on each node of a cluster, so your bootstrap action can call AWS CLI commands.

The following example demonstrates a simple bootstrap action script that copies a file, `myfile.jar`, from Amazon S3 to a local folder, `/mnt1/myfolder`, on each cluster node. The script is saved to Amazon S3 with the file name `copymyfile.sh` with the following contents.

```
aws s3 cp s3://mybucket/myfilefolder/myfile.jar /mnt1/myfolder
```

When you launch the cluster, you specify the script. The following AWS CLI example demonstrates this:

```
aws emr create-cluster --name "Test cluster" --release-label emr-5.20.0 \
--use-default-roles --ec2-attributes KeyName=myKey \
--applications Name=Hive Name=Pig \
--instance-count 3 --instance-type m4.large \
--bootstrap-actions Path="s3://mybucket/myscriptfolder/copymyfile.sh"
```

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: the *master node*, the *core node*, and *task nodes*. Each node type 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 data, transfer output to Amazon S3 or HDFS, and provide status metadata back to the master node. With 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.

When you create a cluster, you make choices that ultimately determine the performance profile of your cluster. This chapter covers those options in detail, and then ties them all together with best practices and guidelines.

Note

The instance fleets configuration is available only in Amazon EMR release versions 4.8.0 and later, excluding 5.0.0 and 5.0.3.

Topics

- [Understanding Master, Core, and Task Nodes \(p. 96\)](#)
- [Configure EC2 Instances \(p. 97\)](#)
- [Configure Networking \(p. 102\)](#)
- [Create a Cluster with Instance Fleets or Uniform Instance Groups \(p. 111\)](#)
- [Cluster Configuration Guidelines and Best Practices \(p. 122\)](#)

Understanding Master, Core, and Task Nodes

Every Amazon EMR cluster has a single master node that runs on an EC2 instance. In addition, you can add capacity to a cluster additional instance groups or instance fleets for core nodes and task nodes. Use this section to understand how Amazon EMR uses each of these node types and as a foundation for cluster capacity planning.

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. Because there is only one master node, the instance group or instance fleet consists of a single EC2 instance.

To monitor the progress of a cluster and interact directly with applications, you can connect to the master node over SSH as the Hadoop user. For more information, see [Connect to the Master Node Using SSH \(p. 239\)](#). Connecting to the master node allows you to access directories and files, such as Hadoop log files, directly. For more information, see [View Log Files \(p. 211\)](#). You can also view user interfaces that applications publish as websites running on the master node. For more information, see [View Web Interfaces Hosted on Amazon EMR Clusters \(p. 243\)](#).

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.

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 \(p. 253\)](#). 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 \(p. 112\)](#).

Warning

Removing HDFS daemons from a running core node or terminating core nodes risks data loss. Use caution when configuring core nodes to use Spot Instances. For more information, see [When Should You Use Spot Instances? \(p. 123\)](#).

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.

Because Spot Instances are often used to run task nodes, Amazon EMR has default functionality for scheduling YARN jobs so that running jobs don't fail when task nodes running on Spot Instances are terminated. Amazon EMR does this by allowing application master processes to run only on core nodes. The application master process controls running jobs and needs to stay alive for the life of the job.

Amazon EMR release version 5.19.0 and later uses the built-in [YARN node labels](#) feature to achieve this. (Earlier versions used a code patch). Properties in the `yarn-site` and `capacity-scheduler` configuration classifications are configured by default so that the YARN capacity-scheduler and fair-scheduler take advantage of node labels. Amazon EMR automatically labels core nodes with the `CORE` label, and sets properties so that application masters are scheduled only on nodes with the `CORE` label. Manually modifying related properties in the `yarn-site` and `capacity-scheduler` configuration classifications, or directly in associated XML files, could break this feature or modify this functionality.

For information about specific properties, see [Amazon EMR Settings To Prevent Job Failure Because of Task Node Spot Instance Termination \(p. 123\)](#).

Configure EC2 Instances

EC2 instances come in different configurations, which are known as *instance types*. Each instance type has different CPU, input/output, and storage capacities. 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, see [Create a Cluster with Instance Fleets or Uniform Instance Groups \(p. 111\)](#). For guidance about choosing the right options, see [Cluster Configuration Guidelines and Best Practices \(p. 122\)](#).

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](#).

Topics

- [Supported Instance Types \(p. 98\)](#)
- [Instance Purchasing Options \(p. 99\)](#)
- [Instance Store and Amazon EBS \(p. 101\)](#)

Supported Instance Types

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

Not all instance types are available in all regions. If you create a cluster using an instance type that is not available, your cluster may fail to provision or may be stuck provisioning. For information about instance availability, see the [Amazon EC2 Pricing Page](#), click the link for your instance purchasing option, and filter by **Region** to see if the instance type you select from the list below is available in the region.

Beginning with Amazon EMR release version 5.13.0, all instances use HVM virtualization and EBS-backed storage for root volumes. When using Amazon EMR release versions earlier than 5.13.0, some previous generation instances use PVM virtualization. These are indicated in the table. For more information, see [Linux AMI Virtualization Types](#).

Some instance types support enhanced networking. For more information, see [Enhanced Networking on Linux](#).

Amazon EMR supports *Previous Generation Instances* to support applications that are optimized for these instances and have not yet been upgraded. For more information about these instance types and upgrade paths, see [Previous Generation Instances](#).

Instance Class	Instance Types
General purpose	<i>m1.medium¹ m1.large¹ m1.xlarge¹ m2.xlarge¹ m2.2xlarge¹ m2.4xlarge¹ m3.xlarge¹ m3.2xlarge¹ m4.large m4.xlarge m4.2xlarge m4.4xlarge m4.10xlarge m4.16xlarge m5.xlarge m5.2xlarge m5.4xlarge m5.12xlarge m5.24xlarge m5a.xlarge m5a.2xlarge m5a.4xlarge m5a.12xlarge m5a.24xlarge m5d.xlarge³ m5d.2xlarge³ m5d.4xlarge³ m5d.12xlarge³ m5d.24xlarge³</i>
Compute optimized	<i>c1.medium¹ c1.xlarge¹ c3.xlarge¹ c3.2xlarge¹ c3.4xlarge¹ c3.8xlarge¹ c4.large c4.xlarge c4.2xlarge c4.4xlarge c4.8xlarge c5.xlarge c5.2xlarge c5.4xlarge c5.9xlarge c5.18xlarge c5d.xlarge³ c5d.2xlarge³ c5d.4xlarge³ c5d.9xlarge³ c5d.18xlarge³ c5n.xlarge c5n.2xlarge c5n.4xlarge c5n.9xlarge c5n.18xlarge cc2.8xlarge z1d.xlarge z1d.2xlarge z1d.3xlarge z1d.6xlarge z1d.12xlarge</i>
Memory optimized	<i>r3.xlarge r3.2xlarge r3.4xlarge r3.8xlarge r4.xlarge r4.2xlarge r4.4xlarge r4.8xlarge r4.16xlarge r5.xlarge³ r5.2xlarge³ r5.4xlarge³ r5.12xlarge³ r5a.xlarge r5a.2xlarge r5a.4xlarge r5a.12xlarge r5a.24xlarge r5d.xlarge³ r5d.2xlarge³ r5d.4xlarge³ r5d.12xlarge³ r5d.24xlarge³ cr1.8xlarge</i>
Storage optimized	<i>h1.2xlarge h1.4xlarge h1.8xlarge h1.8xlarge hs1.8xlarge¹ i2.xlarge i2.2xlarge i2.4xlarge i2.8xlarge i3.xlarge i3.2xlarge i3.4xlarge i3.8xlarge i3.16xlarge d2.xlarge d2.2xlarge d2.4xlarge d2.8xlarge</i>

Instance Class	Instance Types
	Note i3-series instances available when using Amazon EMR version 5.9.0 and later.
GPU instances	<code>cg1.4xlarge g2.2xlarge g3.4xlarge g3.8xlarge g3.16xlarge g3s.xlarge p2.xlarge p2.8xlarge p2.16xlarge p3.2xlarge p3.8xlarge p3.16xlarge</code> Note NVIDIA and CUDA drivers are installed on P2 and P3 instance types by default.

¹Uses PVM virtualization AMI with Amazon EMR release versions earlier than 5.13.0. For more information, see [Linux AMI Virtualization Types](#).

²Not supported in release version 5.15.0.

³Supported in release version 5.13.0 and later.

Instance Purchasing Options

When you set up a cluster, you choose a purchasing option for EC2 instances. You can choose to use On-Demand Instances, Spot Instances, or both. Prices vary based on the instance type and region. For current pricing, see [Amazon EMR Pricing](#).

Your choice to use instance groups or instance fleets in your cluster determines how you can change instance purchasing options while a cluster is running. If you choose uniform instance groups, the instance type and purchasing option apply to all EC2 instances in each instance group, and you can only specify the purchasing option for an instance group when you create it. If you choose instance fleets, you can change purchasing options after you create the instance fleet, 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 \(p. 111\)](#).

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

With On-Demand Instances, you pay for compute capacity by the hour. Optionally, you can have these On-Demand Instances use Reserved Instance or Dedicated Instance purchasing options. With Reserved Instances, you make a one-time payment for an instance to reserve capacity. Dedicated Instances are physically isolated at the host hardware level from instances that belong to other AWS accounts. For more information about purchasing options, 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 purchase a Reserved Instance, if all of the following conditions are true, Amazon EMR uses the Reserved Instance when a cluster launches:

- 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 `m4.1large` Reserved Instance with the instance reservation scoped to the US-East region. You then launch an EMR cluster in US-East that uses two `m4.1large` 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 match the 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. For more information about when using Spot Instances may be appropriate for your application, see [When Should You Use Spot Instances? \(p. 123\)](#).

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. When you choose Spot Instances, you specify the maximum Spot price that you're willing to pay for each EC2 instance type. When the Spot price in the cluster's Availability Zone is below the maximum Spot price specified for that instance type, the instances launch. While instances run, you're charged at the current Spot price *not your maximum Spot price*.

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 are not interrupted before the duration expires. This topic describes how Spot Instances work with Amazon EMR.

For current pricing, see [Amazon EC2 Spot Instances Pricing](#). For more information, see [Spot Instances](#) in the *Amazon EC2 User Guide for Linux Instances*. 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 [Configure Networking \(p. 102\)](#).

Tip

You can see the real-time Spot price in the console when you hover over the information tooltip next to the **Spot** purchasing option when you create a cluster using **Advanced Options**. The prices for each Availability Zone in the selected region are displayed. The lowest prices are in the green-colored rows. 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 maximum Spot price. Spot Instances of that type launch if the Spot price in the cluster's region and Availability Zone is below the maximum Spot price. Instances terminate if the Spot price is above your maximum Spot price. You set the maximum Spot 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 \(p. 111\)](#).

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 \(p. 112\)](#).

Instance Store and Amazon EBS

There are two types of storage volumes available for EC2 instances: Amazon EBS volumes and the instance store. Amazon EBS volumes are available only in Amazon EMR version 4.0 and later. Whether the root device volume uses the instance store or an Amazon 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 EBS works differently within Amazon EMR than it does with regular Amazon EC2 instances. Amazon 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's important that you not expect data to persist. Although the data is ephemeral, 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 storage volumes, these are mounted as additional volumes. They are not a part of the boot volume. 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).

Instance store and/or EBS volume storage is used for HDFS data, as well as buffers, caches, scratch data, and other temporary content that some applications may "spill" to the local file system. EMRFS can help ensure that there is a persistent "source of truth" for HDFS data stored in Amazon S3.

Amazon EMR automatically attaches an Amazon 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 Amazon EBS charges for gp2 volumes in the region where the cluster runs. For example, the EBS cost per hour for the root volume on each cluster node 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.

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 can't snapshot an EBS volume and then restore it within Amazon EMR. To create reusable custom configurations, use a custom AMI (available in Amazon EMR version 5.7.0 and later). For more information, see [Using a Custom AMI \(p. 85\)](#).
- An encrypted EBS root storage volume is supported only when using a custom AMI. For more information, see [Creating a Custom AMI with an Encrypted Amazon EBS Root Device Volume \(p. 88\)](#). Encrypted EBS storage volumes are not supported.
- If you apply tags using the Amazon EMR API, those operations are applied to EBS volumes.
- There is a limit of 25 volumes per instance.

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 \(p. 103\)](#)
- [Set up a VPC to Host Clusters \(p. 106\)](#)
- [Launch Clusters into a VPC \(p. 108\)](#)
- [Restrict Permissions to a VPC Using IAM \(p. 109\)](#)
- [Minimum Amazon S3 Policy for Private Subnet \(p. 110\)](#)
- [More Resources for Learning About VPCs \(p. 110\)](#)

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

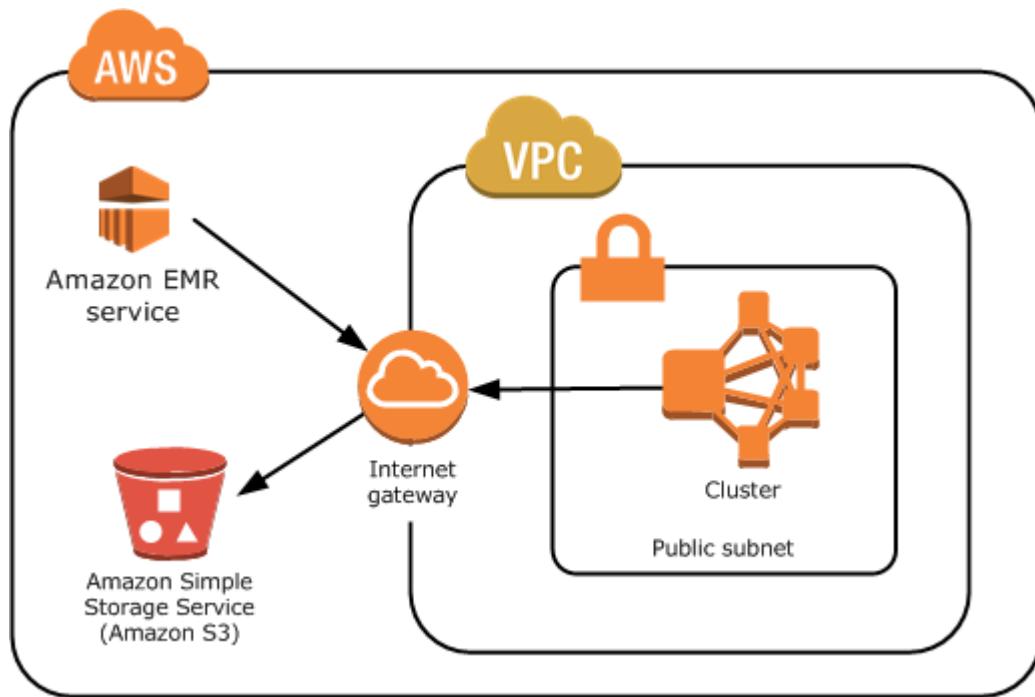
EMR clusters in a public subnet require a connected internet gateway. This is because Amazon EMR clusters must access AWS services and Amazon EMR. If a service, such as Amazon S3, provides the ability to create a VPC endpoint, you can access those services using the endpoint instead of accessing a public 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.

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.

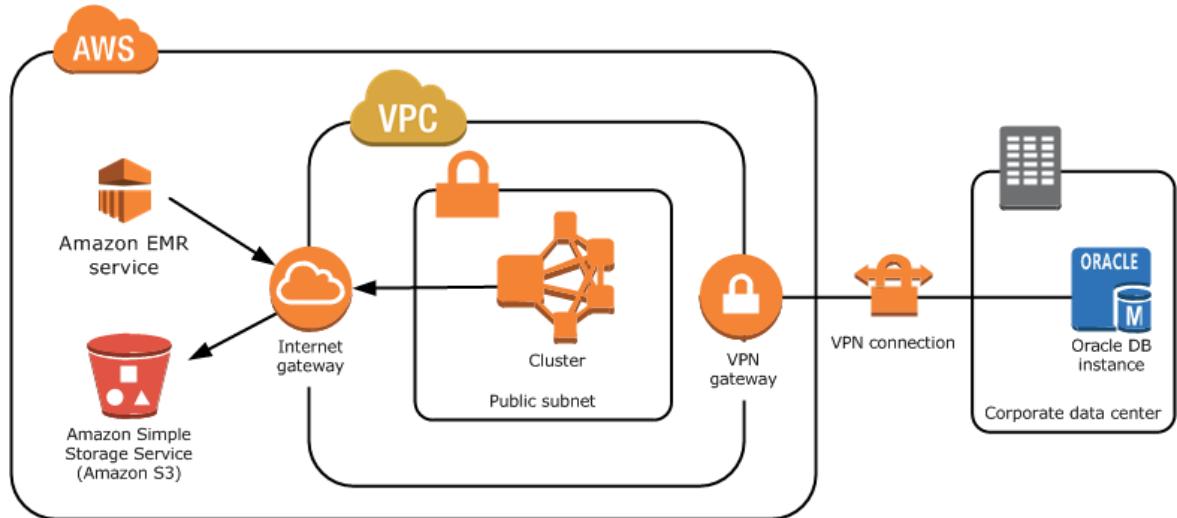
If you have additional AWS resources that you do not want connected to the internet gateway, you can launch those components in a private subnet that you create within your VPC.

Clusters running in a public subnet use two security groups: one for the master node and another for core and task nodes. For more information, see [Control Network Traffic with Security Groups \(p. 168\)](#).

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 backend. 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.

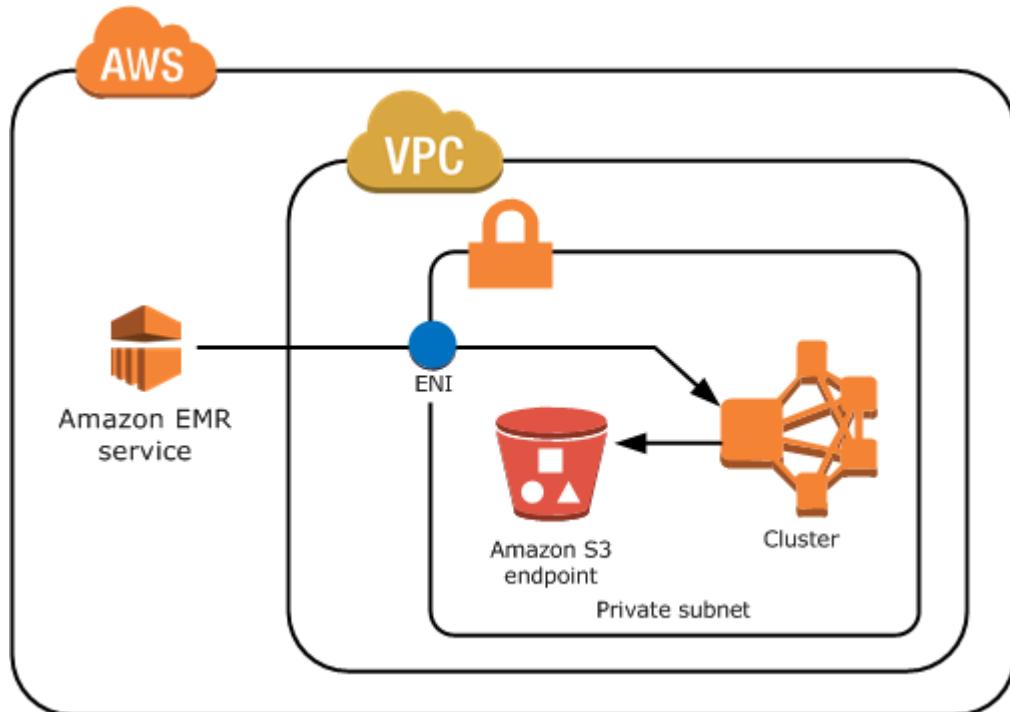
The following are differences from public subnets:

- To access AWS services that do not provide a VPC endpoint, you still must use a NAT instance or an internet gateway.
- 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 \(p. 110\)](#)
- 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, see [NAT Gateways](#) in the [Amazon VPC 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.

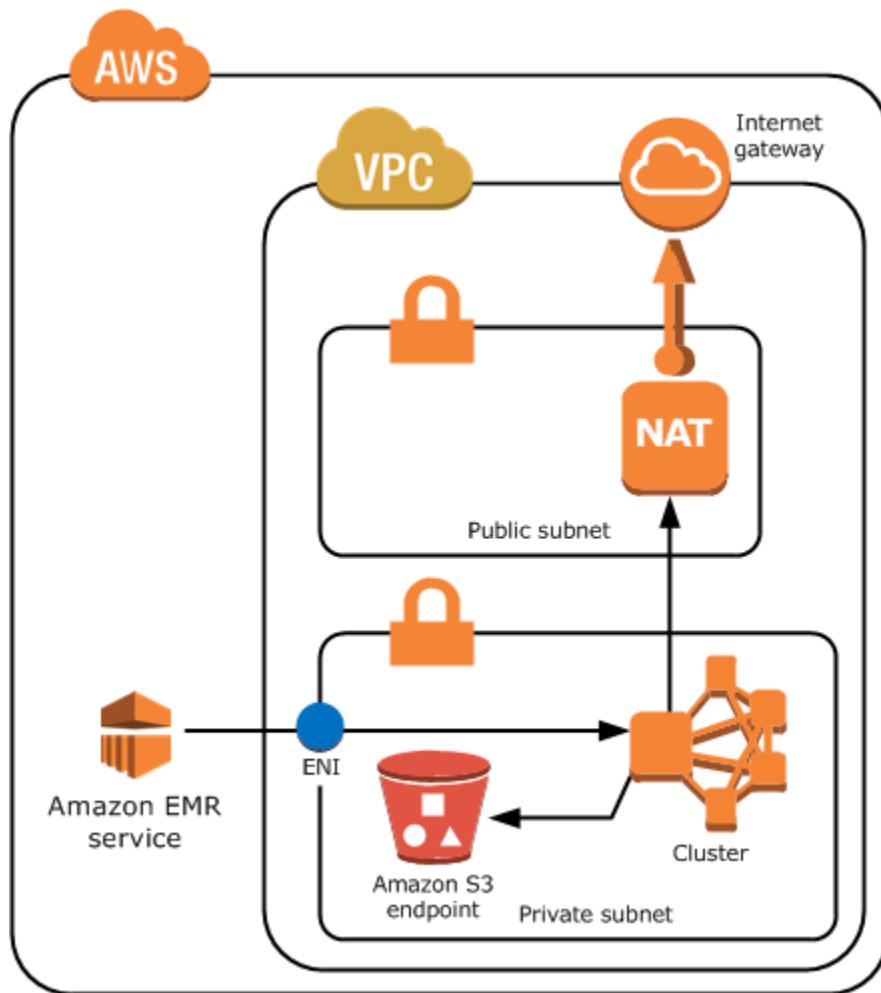
Amazon EMR creates and uses different default security groups for the clusters in a private subnet: ElasticMapReduce-Master-Private, ElasticMapReduce-Slave-Private, and ElasticMapReduce-ServiceAccess. For more information, see [Control Network Traffic with Security Groups \(p. 168\)](#).

For a complete listing of NACLs of your cluster, choose **Security groups for Master** and **Security groups for Core & Task** on 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 that is 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: * <input type="text" value="10.0.0.0/16"/> (65531 IP addresses available)	
VPC name: <input type="text" value="My VPC"/>	
Public subnet: * <input type="text" value="10.0.0.0/24"/> (251 IP addresses available) Availability Zone: * <input type="text" value="No Preference"/> <input type="button" value="▼"/> Subnet name: <input type="text" value="Public subnet"/> <small>You can add more subnets after AWS creates the VPC.</small>	
Enable DNS hostnames: * <input checked="" type="radio"/> Yes <input type="radio"/> No Hardware tenancy: * <input type="text" value="Default"/> <input type="button" value="▼"/>	
<input type="button" value="Cancel and Exit"/> <input type="button" value="Back"/> <input style="background-color: #0072BC; color: white; font-weight: bold; border-radius: 5px; padding: 5px 10px; border: none;" type="button" value="Create VPC"/>	

Step 2: VPC with Public and Private Subnets

IP CIDR block: * <input type="text" value="10.0.0.0/16"/> (65531 IP addresses available)	
VPC name: <input type="text"/>	
Public subnet: * <input type="text" value="10.0.0.0/24"/> (251 IP addresses available) Availability Zone: * <input type="text" value="No Preference"/> <input type="button" value="▼"/> Public subnet name: <input type="text" value="Public subnet"/>	
Private subnet: * <input type="text" value="10.0.1.0/24"/> (251 IP addresses available) Availability Zone: * <input type="text" value="No Preference"/> <input type="button" value="▼"/> Private subnet name: <input type="text" value="Private subnet"/> <small>You can add more subnets after AWS creates the VPC.</small>	
Specify the details of your NAT instance <small>(Instance rates apply).</small> Instance type: * <input type="text" value="m1.small"/> <input type="button" value="▼"/> Key pair name: <input type="text" value="No key pair"/> <input type="button" value="▼"/>	
Add endpoints for S3 to your subnets Subnet: <input type="text" value="None"/> <input type="button" value="▼"/>	
Enable DNS hostnames: * <input checked="" type="radio"/> Yes <input type="radio"/> No Hardware tenancy: * <input type="text" value="Default"/> <input type="button" value="▼"/>	

- 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, all EC2 instances must at minimum have a route to Amazon EMR through the elastic network interface. 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-West) 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. For more information, see [Control Network Traffic with Security Groups \(p. 168\)](#).

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 VPCCLI 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 m4.large --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": "AmazonLinuxAMIRRepositoryAccess",  
            "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/*"  
            ]  
        }  
    ]  
}
```

More Resources for Learning About VPCs

Use the following topics to learn more about VPCs and subnets.

- 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 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.

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 \(p. 112\)](#).

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 \(p. 120\)](#). For information about adding and removing instances, see [Scaling Cluster Resources \(p. 253\)](#).

Topics

- [Configure Instance Fleets \(p. 112\)](#)
- [Configure Uniform Instance Groups \(p. 120\)](#)

Configure Instance Fleets

The instance fleets configuration for a cluster offers the widest variety of provisioning options for EC2 instances. With instance fleets, you specify *target capacities* for On-Demand Instances and Spot Instances within each fleet. When the cluster launches, Amazon EMR provisions instances until the targets are fulfilled. You can specify up to five EC2 instance types per fleet for Amazon EMR to use when fulfilling the targets. You can also select multiple subnets for different Availability Zones. When Amazon EMR launches the cluster, it looks across those subnets to find the instances and purchasing options you specify.

While a cluster is running, if Amazon EC2 reclaims a Spot Instance because of a price increase, or an instance fails, Amazon EMR tries to replace the instance with any of the instance types that you specify. This makes it easier to regain capacity during a spike in Spot pricing. Instance fleets allow you to 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, and then switch to On-Demand capacity if Spot isn't available at your price.

Note

The instance fleets configuration is available only in Amazon EMR release versions 4.8.0 and later, excluding 5.0.0 and 5.0.3.

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 Instances for the core fleet and task fleet. Use vCPU or a generic unit assigned to each EC2 instance that counts toward the targets. Amazon EMR provisions instances until each target capacity is totally fulfilled. For the master fleet, the target is always one.
- 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 Spot price. Amazon EMR provisions Spot Instances if the Spot price is below the maximum Spot price. You pay the Spot price, not necessarily the maximum Spot price.
 - Optionally, specify a defined duration (also known as a Spot block) for each fleet. Spot Instances terminate only after the defined duration expires.
 - For each fleet, define a timeout period for provisioning Spot Instances. If Amazon EMR can't provision Spot capacity, you can terminate the cluster or switch to provisioning On-Demand capacity instead.

Instance Fleet Options

Use the following guidelines to understand instance fleet options.

Setting Target Capacities

Specify the target capacities you want for the core fleet and task fleet. When you do, that determines the number of On-Demand Instances and Spot Instances that Amazon EMR provisions. When you specify an

instance, you decide how much each instance counts toward the target. When an On-Demand Instance is provisioned, it counts toward the On-Demand target. The same is true for Spot Instances. Unlike core and task fleets, the master fleet is always one instance. Therefore, the target capacity for this fleet is always one.

When you use the console, the vCPUs of the EC2 instance type are used as the count for target capacities by default. You can change this to **Generic units**, and then specify the count for each EC2 instance type. When you use the AWS CLI, you manually assign generic units for each instance 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](#).

For each fleet, you specify up to five EC2 instance types. Amazon EMR chooses any combination of these EC2 instance types to fulfill your target capacities. Because Amazon EMR wants to fill target capacity completely, an overage might happen. For example, if there are two unfulfilled units, and Amazon EMR can only provision an instance with a count of five units, the instance still gets provisioned, meaning that the target capacity is exceeded by three units.

If you reduce the target capacity to resize a running cluster, Amazon EMR attempts to complete application tasks and terminates instances to meet the new target. For more information, see [Terminate at Task Completion \(p. 268\)](#). Amazon EMR has a 60-minute timeout for completing a resize operation. In some cases, a node may still have tasks running after 60 minutes, and Amazon EMR reports that the resize operation was successful and that the new target was not met.

Spot Instance Options

You specify a **Maximum Spot price** for each of the five instance types in a fleet. You can set this price either as a percentage of the On-Demand price, or as a specific dollar amount. Amazon EMR provisions Spot Instances if the current Spot price in an Availability Zone is below your maximum Spot price. You pay the Spot price, not necessarily the maximum Spot price.

You can specify a **Defined duration** for Spot Instances in a fleet. When the Spot price changes, Amazon EMR doesn't terminate instances until the **Defined duration** expires. Defined duration pricing applies when you select this option. If you don't specify a defined duration, instances terminate as soon as the Spot price exceeds the maximum Spot price. For more information, see [Specifying a Duration for Your Spot Instances](#) and [Amazon EC2 Spot Instances Pricing](#) for defined duration pricing.

For each fleet, you also define a **Provisioning timeout**. The timeout applies when the cluster is provisioning capacity when it is created and can't provision enough Spot Instances to fulfill target capacity according to your specifications. You specify the timeout period and the action to take. You can have the cluster terminate or switch to provisioning On-Demand capacity to fulfill the remaining Spot capacity. When you choose to switch to On-Demand, the remaining Spot capacity is effectively added to the On-Demand target capacity after the timeout expires.

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

Multiple Subnet (Availability Zones) Options

When you use instance fleets, 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 identifies the best Availability Zone to launch instances according to your fleet specifications. 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.

Use the Console to Configure Instance Fleets

To create a cluster using instance fleets, use the **Advanced options** configuration in the Amazon EMR console.

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. For **Network**, enter a value. If you choose a 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** that 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 Spot price** options. You can enter your maximum Spot price as a **% of On-Demand** pricing, or you can enter a **Dollars (\$)** amount in USD.

Tip

Hover over the information tooltip for **Maximum Spot 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 Spot 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**, modify other cluster settings, and then 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 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=m4.large}' ]
  \
  InstanceFleetType=CORE,TargetOnDemandCapacity=1,InstanceTypeConfigs=[ '{InstanceType=m4.large}' ]
```

Example 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=m4.large,BidPrice=0.5}' ]
  \
  InstanceFleetType=CORE,TargetSpotCapacity=1,InstanceTypeConfigs=[ '{InstanceType=m4.large,BidPrice=0.5}' ]
```

Example 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=m4.large}' ]
  \
  InstanceFleetType=CORE,TargetOnDemandCapacity=2,TargetSpotCapacity=6,InstanceTypeConfigs=[ '{InstanceType=m4.large,BidPrice=0.5}' ]
```

Example 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=m4.large}' ]
  \
  InstanceFleetType=CORE,TargetSpotCapacity=11,InstanceTypeConfigs=[ '{InstanceType=m4.large,BidPrice=0.5,' \
'{InstanceType=m4.2xlarge,BidPrice=0.9,WeightedCapacity=5}'], \
LaunchSpecifications={SpotSpecification='{TimeoutDurationMinutes=120,TimeoutAction=SWITCH_TO_ON_DEMAND}'}
```

Example 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=m4.large}' ]
  \
  InstanceFleetType=CORE,TargetOnDemandCapacity=8,TargetSpotCapacity=6, \
  InstanceTypeConfigs=[ '{InstanceType=m4.large,BidPrice=0.5,WeightedCapacity=3}', \
'{InstanceType=m4.2xlarge,BidPrice=0.9,WeightedCapacity=5}'], \
  LaunchSpecifications={SpotSpecification='{TimeoutDurationMinutes=120,TimeoutAction=SWITCH_TO_ON_DEMAND}'}
  \
  InstanceFleetType=TASK,TargetOnDemandCapacity=3,TargetSpotCapacity=3,
```

```
InstanceTypeConfigs=['{InstanceType=m4.large,BidPrice=0.5,WeightedCapacity=3}']
```

Example 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=m4.large,BidPrice=0.5, \
EbsConfiguration={EbsOptimized=true,EbsBlockDeviceConfigs=[{VolumeSpecification={VolumeType=gp2, \
SizeIn GB=100}},{VolumeSpecification={VolumeType=io1,SizeInGB=100,Iop \
s=100},VolumesPerInstance=4}]}']
```

Example 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 maximum Spot price (`BidPrice`) as a percentage of on-demand, while the task and master instance fleets use a maximum Spot price (`BidPriceAsPercentageofOnDemandPrice`) as a string in USD.

```
[ \
  { \
    "Name": "Masterfleet", \
    "InstanceFleetType": "MASTER", \
    "TargetSpotCapacity": 1, \
    "LaunchSpecifications": { \
      "SpotSpecification": { \
        "TimeoutDurationMinutes": 120, \
        "TimeoutAction": "SWITCH_TO_ON_DEMAND" \
      } \
    }, \
    "InstanceTypeConfigs": [ \
      { \
        "InstanceType": "m4.large", \
        "BidPrice": "0.89" \
      } \
    ] \
  }, \
  { \
    "Name": "Corefleet", \
    "InstanceFleetType": "CORE", \
    "TargetSpotCapacity": 1, \
    "LaunchSpecifications": { \
      "SpotSpecification": { \
        "TimeoutDurationMinutes": 120, \
        "TimeoutAction": "TERMINATE_CLUSTER" \
      } \
    } \
  }]
```

```

        },
        "InstanceTypeConfigs": [
            {
                "InstanceType": "m4.large",
                "BidPriceAsPercentageOfOnDemandPrice": 100
            }
        ]
    },
    {
        "Name": "Taskfleet",
        "InstanceFleetType": "TASK",
        "TargetSpotCapacity": 1,
        "LaunchSpecifications": {
            "SpotSpecification": {
                "TimeoutDurationMinutes": 120,
                "TimeoutAction": "TERMINATE_CLUSTER"
            }
        },
        "InstanceTypeConfigs": [
            {
                "InstanceType": "m4.large",
                "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-12ABCDEFHGI34JK'
```

```

{
    "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,
            "ProvisionedOnDemandCapacity": 2
        }
    ]
}

```

```

    "InstanceTypeSpecifications": [
        {
            "BidPrice": "0.5",
            "InstanceType": "m4.large",
            "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": "m4.large",
            "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-12ABCDEFHI34JK' /
--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-12ABCDEFHI34JK' --instance-fleet
    InstanceFleetType=TASK,TargetSpotCapacity=1,/
    LaunchSpecifications={SpotSpecification='{TimeoutDurationMinutes=20,TimeoutAction=TERMINATE_CLUSTER}'},
    InstanceTypeConfigs=['{InstanceType=m4.large,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. 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. You can choose different instance types and purchasing options for each instance group. For more information, see [Scaling Cluster Resources \(p. 253\)](#).

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 \(p. 203\)](#).

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** for your cluster. The subnet that you choose is associated with an Availability Group, which is listed with each subnet. For more information, see [Configure Networking \(p. 102\)](#).

Note

Your account and region may give you the option to choose **Launch into EC2-Classic 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. 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 to use for each node type. There is only one instance in the **Master** node type.
- Under **Purchasing option**, choose **On-demand** or **Spot**. If you choose **Spot**, select an option for the maximum price for Spot Instances. By default, **Use on-demand as max price** is selected. You can select **Set max \$/hr** and then enter your maximum price. Availability Zone of the **EC2 Subnet** you chose is below the **Maximum Spot price**.

Tip

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

- Under **Auto Scaling for Core and Task node types**, choose the pencil icon, and then configure the automatic scaling options. For more information, see [Using Automatic Scaling in Amazon EMR \(p. 254\)](#).
7. Choose **Add task instance group** as desired and configure settings as described in the previous step.
 8. Choose **Next**, modify other cluster settings, and then 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 Command 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 that 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

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 "MySpotCluster" --release-label emr-5.20.0 \
--use-default-roles --ec2-attributes KeyName=myKey \
--instance-groups
  InstanceGroupType=MASTER,InstanceType=m4.large,InstanceCount=1,BidPrice=0.25 \
  InstanceGroupType=CORE,InstanceType=m4.large,InstanceCount=2,BidPrice=0.03 \
  InstanceGroupType=TASK,InstanceType=m4.large,InstanceCount=4,BidPrice=0.03 \
  InstanceGroupType=TASK,InstanceType=m4.large,InstanceCount=2,BidPrice=0.04
```

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("m4.large")
  .withMarket("SPOT")
  .withBidPrice("0.25");
```

```
InstanceGroupConfig instanceGroupConfigCore = new InstanceGroupConfig()
    .withInstanceCount(4)
    .withInstanceRole("CORE")
    .withInstanceType("m4.large")
    .withMarket("SPOT")
    .withBidPrice("0.03");

InstanceGroupConfig instanceGroupConfigTask = new InstanceGroupConfig()
    .withInstanceCount(2)
    .withInstanceRole("TASK")
    .withInstanceType("m4.large")
    .withMarket("SPOT")
    .withBidPrice("0.10");
```

Cluster Configuration Guidelines and Best Practices

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 a cluster, which depend on whether you use the instance groups configuration or the instance fleets configuration for the cluster.

- **Instance Groups**
 - Manually add instances of the same type to existing core and task instance groups.
 - Manually add a task instance group, which can use a different instance type.
 - Set up automatic scaling in Amazon EMR for an instance group, adding and removing instances automatically based on the value of an Amazon CloudWatch metric that you specify. For more information, see [Scaling Cluster Resources \(p. 253\)](#).
- **Instance Fleets**
 - Add a single task instance fleet.
 - Change the target capacity for On-Demand and Spot Instances for existing core and task instance fleets. For more information, see [Configure Instance Fleets \(p. 112\)](#).

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 \(p. 203\)](#). 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 [Configure EC2 Instances \(p. 97\)](#).

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 m4.large instance. For clusters of more than 50 nodes, consider using an m4.xlarge.
- The computational needs of the core and task nodes depend on the type of processing your application performs. Many jobs can be run on m4.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 t2.medium instances to reduce costs while the instances are waiting for dependencies to finish. For improved performance, consider running the cluster using m4.xlarge 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 m4.large and m4.xlarge. 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 datasets 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 a limit increase for your account, see [AWS Limits](#).

When Should You Use Spot Instances?

When you launch a cluster in Amazon EMR, you can choose to launch master, core, or task instances on Spot Instances. Because each type of instance group plays a different role in the cluster, there are implications of launching each node type on Spot Instances. You can't change an instance purchasing option while a cluster is running. To change from On-Demand to Spot Instances or vice versa, for the master and core nodes, you must terminate the cluster and launch a new one. For task nodes, you can launch a new task instance group or instance fleet, and remove the old one.

Topics

- [Amazon EMR Settings To Prevent Job Failure Because of Task Node Spot Instance Termination \(p. 123\)](#)
- [Master Node on a Spot Instance \(p. 124\)](#)
- [Core Nodes on Spot Instances \(p. 124\)](#)
- [Task Nodes on Spot Instances \(p. 124\)](#)
- [Instance Configurations for Application Scenarios \(p. 125\)](#)

Amazon EMR Settings To Prevent Job Failure Because of Task Node Spot Instance Termination

Because Spot Instances are often used to run task nodes, Amazon EMR has default functionality for scheduling YARN jobs so that running jobs don't fail when task nodes running on Spot Instances are terminated. Amazon EMR does this by allowing application master processes to run only on core nodes. The application master process controls running jobs and needs to stay alive for the life of the job.

Amazon EMR release version 5.19.0 and later uses the built-in [YARN node labels](#) feature to achieve this. (Earlier versions used a code patch). Properties in the `yarn-site` and `capacity-scheduler` configuration classifications are configured by default so that the YARN capacity-scheduler and fair-scheduler take advantage of node labels. Amazon EMR automatically labels core nodes with the `CORE` label, and sets properties so that application masters are scheduled only on nodes with the `CORE` label. Manually modifying related properties in the `yarn-site` and `capacity-scheduler` configuration classifications, or directly in associated XML files, could break this feature or modify this functionality.

Amazon EMR configures the following properties and values by default. Use caution when configuring these properties.

- **yarn-site (yarn-site.xml) On All Nodes**

- `yarn.node-labels.enabled: true`
- `yarn.node-labels.am.default-node-label-expression: 'CORE'`
- `yarn.node-labels.fs-store.root-dir: '/apps/yarn/nodelabels'`
- `yarn.node-labels.configuration-type: 'distributed'`
- **yarn-site (yarn-site.xml) On Master And Core Nodes**
- `yarn.nodemanager.node-labels.provider: 'config'`
- `yarn.nodemanager.node-labels.provider.configured-node-partition: 'CORE'`
- **capacity-scheduler (capacity-scheduler.xml) On All Nodes**
- `yarn.scheduler.capacity.root.accessible-node-labels: '*'`
- `yarn.scheduler.capacity.root.accessible-node-labels.CORE.capacity: 100`
- `yarn.scheduler.capacity.root.default.accessible-node-labels: '*'`
- `yarn.scheduler.capacity.root.default.accessible-node-labels.CORE.capacity: 100`

Master Node on a 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 consider when selecting your maximum Spot 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 Nodes on Spot Instances

Core nodes process data and store information using HDFS. Terminating a core instance risks data loss. For this reason, you should only run core nodes on Spot Instances when partial HDFS data loss is tolerable.

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. In other words, if you request six Amazon EC2 instances, and only five are available at or below your maximum Spot price, the instance group won't launch. Amazon EMR continues to wait until all six Amazon EC2 instances are available or until you terminate the cluster. You can change the number of Spot Instances in a core instance group to add capacity to a running cluster. For more information about working with instance groups, and how Spot Instances work with instance fleets, see [the section called "Configure Instance Fleets or Instance Groups" \(p. 111\)](#).

Task Nodes on 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 maximum Spot 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, using your maximum Spot price. This means that if you request a task instance

group with six nodes, and only five Spot Instances are available at or below your maximum Spot price, Amazon EMR launches the instance group with five nodes, adding the sixth later if possible.

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. You can add task instances to your task instance groups as needed, to handle peak traffic or speed up data processing.

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

Instance Configurations for Application Scenarios

The following table is a quick reference to node type purchasing options and configurations that are usually appropriate for various application scenarios. Choose the link to view more information about each scenario type.

Application Scenario	Master Node Purchasing Option	Core Nodes Purchasing Option	Task Nodes Purchasing Option
Long-Running Clusters and Data Warehouses (p. 125)	On-Demand	On-Demand or instance-fleet mix	Spot or instance-fleet mix
Cost-Driven Workloads (p. 125)	Spot	Spot	Spot
Data-Critical Workloads (p. 125)	On-Demand	On-Demand	Spot or instance-fleet mix
Application Testing (p. 125)	Spot	Spot	Spot

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 that has a predictable variation in computational capacity, such as a data warehouse, you can handle peak demand at lower cost with Spot Instances. You can launch your master and core instance groups as On-Demand Instances 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.

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 used for core nodes.
- The capacity of the EC2 instance store for the instance type used. For more information on instance store volumes, see [Amazon EC2 Instance Store](#) in the *Amazon EC2 User Guide for Linux 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, two for a cluster of 4-9 core nodes, and one for a cluster of three or fewer nodes.

To calculate the HDFS capacity of a cluster, for each core node, add the instance store volume capacity to the EBS storage capacity (if used). Multiply the result by the number of core nodes, and then divide the total by the replication factor based on the number of core nodes. For example, a cluster with 10 core nodes of type i2.xlarge, which have 800 GB of instance storage without any attached EBS volumes, has a total of approximately 2,666 GB available for HDFS ($10 \text{ nodes} \times 800 \text{ GB} \div 3 \text{ replication factor}$).

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 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 \(p. 239\)](#). 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 \(p. 211\)](#).

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 \(p. 211\)](#).

6. Proceed with creating the cluster as described in [Plan and Configure Clusters \(p. 39\)](#).

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 m4.large --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 <https://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 m4.large --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 <https://docs.aws.amazon.com/cli/latest/reference/emr>.

Enable the Debugging Tool

The debugging tool allows you to more easily browse log files from the EMR console. For more information, see [View Log Files in the Debugging Tool \(p. 214\)](#). 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 console to browse the step, job, task, and task-attempt logs for the cluster in an intuitive way.

To use the debugging tool in the EMR console, you must enable debugging when you launch the cluster 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 \(p. 39\)](#).

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 m4.large --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 <https://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 Permissions to AWS Services \(p. 187\)](#).

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:

- Restrictions that apply to Amazon EC2 resources apply to Amazon EMR as well. For more information, see https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/Using_Tags.html#tag-restrictions.
- 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 EC2 API 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 m4.large --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 <https://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 <https://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-KT4XXXXXXXXX1NM` with your cluster ID.

```
aws emr describe-cluster --cluster-id j-KT4XXXXXXXXX1NM --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 <https://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 <https://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 \(p. 134\)](#)

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:

- <http://awssupportdatasvcs.com/bootstrap-actions/Simba/AmazonHiveJDBC-1.0.9.1060.zip>

- http://awssupportdatasvcs.com/bootstrap-actions/Simba/Hive_ODBC_1.2.2.1018.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.

Security

Amazon EMR provides several features to help secure cluster resources and data:

- **AWS Identity and Access Management (IAM) policies**

IAM policies allow or deny permissions for IAM users and groups to perform actions. Policies can be combined with tagging to control access on a cluster-by-cluster basis. For more information, see [Use IAM Policies to Allow and Deny User Permissions \(p. 137\)](#).

- **IAM roles for EMRFS requests to Amazon S3**

You can control whether cluster users can access files from within Amazon EMR based on user, group, or the location of EMRFS data in Amazon S3. For more information, see [Configure IAM Roles for EMRFS Requests to Amazon S3 \(p. 164\)](#).

- **IAM roles**

The Amazon EMR service role, instance profile, and service-linked role control how Amazon EMR is able to access other AWS services. For more information, see [Configure IAM Roles for Amazon EMR Permissions to AWS Services \(p. 187\)](#).

- **Kerberos**

You can set up Kerberos to provide strong authentication through secret-key cryptography. For more information, see [Use Kerberos Authentication \(p. 144\)](#).

- **Secure Socket Shell (SSH)**

SSH provides a secure way for users to connect to the command line on cluster instances. It also provides tunneling to view web interfaces that applications host on the master node. Clients can authenticate using Kerberos or an Amazon EC2 key pair. For more information, see [Use an Amazon EC2 Key Pair for SSH Credentials \(p. 157\)](#) and [Connect to the Cluster \(p. 238\)](#).

- **Data encryption**

You can implement data encryption to help protect data at rest in Amazon S3 and in cluster instance storage, and data in transit. For more information, see [Encrypt Data in Transit and At Rest \(p. 157\)](#). You can also use a custom Amazon Linux AMI to encrypt the EBS root device volume of cluster instances. For more information, see [Using a Custom AMI \(p. 85\)](#).

- **Security groups**

Security groups act as a virtual firewall for Amazon EMR cluster instances, limiting inbound and outbound network traffic. For more information, see [Control Network Traffic with Security Groups \(p. 168\)](#).

- **Security configurations**

Security configurations in Amazon EMR are templates for a security setup. You can create a security configuration to conveniently re-use a security setup whenever you create a cluster. For more information, see [Use Security Configurations to Set Up Cluster Security \(p. 175\)](#).

When an Amazon EC2 instance in a cluster that is based on the default Amazon Linux AMI for Amazon EMR boots for the first time, Critical security updates are installed by default. Other updates are not installed. Depending on the security posture of your application and the length of time that a cluster runs, you may choose to periodically reboot your cluster to apply security updates, or create a bootstrap action to customize package installation and updates. You may also choose to test and then install select

security updates on running cluster instances. For more information, see [Working with Amazon Linux AMIs in Amazon EMR \(p. 83\)](#).

Use IAM Policies to Allow and Deny User Permissions

Amazon EMR supports AWS Identity and Access Management (IAM) policies. IAM is a web service that enables AWS customers to manage users and their permissions. You can use IAM to create policies and attach them to principals, such as users and groups. 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 and the AWS CLI. You can also use it 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 \(p. 137\)](#)
- [Use Managed Policies for User Access \(p. 138\)](#)
- [Use Inline Policies for User Permissions \(p. 140\)](#)
- [Use Cluster Tagging with IAM Policies for Cluster-Specific Control \(p. 140\)](#)

Amazon EMR Actions in User-Based IAM Policies

In IAM user 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 \(p. 138\)](#).

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 Tagging with IAM Policies for Cluster-Specific Control \(p. 140\)](#). Because Amazon EMR does not support resource-based or resource-level policies, the Resource element always has a wildcard value.

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. If you use inline policies, service changes may occur that cause permission errors to appear.

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. To create custom policies, we recommend 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.

The contents of Version 6 of this policy are shown below. 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": [  
    {  
      "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>ModifyInstanceState",  
        "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"
    ],
    "Effect": "Allow",
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": "iam>CreateServiceLinkedRole",
    "Resource": "*",
    "Condition": {
        "StringLike": {
            "iam:AWSPropertyName": [
                "elasticmapreduce.amazonaws.com",
                "elasticmapreduce.amazonaws.com.cn"
            ]
        }
    }
}
]
}

```

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 Inline Policies for User Permissions

To customize policies, we recommend that you start with a managed policy and then modify permissions and conditions according to your requirements.

Important

Inline policies are not automatically updated when service requirements change. If you create and attach inline policies, be aware that service updates might occur that suddenly cause permissions errors. For more information, see [Managed Policies and Inline Policies](#) in the *IAM User Guide* and [Specify Custom IAM Roles When You Create a Cluster \(p. 195\)](#).

The `AmazonElasticMapReduceFullAccess`, which is the default managed policy for users to have full permissions for Amazon EMR, includes a statement that allows the `iam:PassRole` permissions for all resources. This statement allows the user to pass any role to other AWS services so that Amazon EMR can interact with those services on behalf of the user.

To implement a more restrictive policy, attach an inline policy to appropriate users or groups that allows `iam:PassRole` only for roles specific to Amazon EMR. The following example demonstrates a statement that allows `iam:PassRole` permissions only for the default Amazon EMR roles: `EMR_DefaultRole`, `EMR_EC2_DefaultRole`, and `EMR_AutoScalingDefaultRole`. If you use custom roles, replace the default role names with your custom role names.

```
{
    "Action": "iam:PassRole",
    "Effect": "Allow",
    "Resource": [
        "arn:aws:iam::*:role/EMR_DefaultRole",
        "arn:aws:iam::*:role/EMR_EC2_DefaultRole",
        "arn:aws:iam::*:role/EMR_AutoScaling_DefaultRole"
    ]
}
```

For more information about roles for Amazon EMR, see [Configure IAM Roles for Amazon EMR Permissions to AWS Services \(p. 187\)](#).

Use Cluster Tagging with IAM Policies for Cluster-Specific 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 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 where `ClusterID` is a required request parameter. For example, the `ModifyInstanceGroups` action does not support context keys because `ClusterID` is an optional parameter.

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 `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": "Stmt12345678901234",
            "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": [
            "*"
        ]
    }
}
```

Use Kerberos Authentication

Amazon EMR release version 5.10.0 and later supports Kerberos, which is a network authentication protocol created by the Massachusetts Institute of Technology (MIT). Kerberos uses secret-key cryptography to provide strong authentication so that passwords or other credentials aren't sent over the network in an unencrypted format.

In Kerberos, services and users that need to authenticate are known as *principals*. Principals exist within a Kerberos *realm*. Within the realm, a Kerberos server known as the *key distribution center (KDC)* provides the means for principals to authenticate. The KDC does this by issuing *tickets* for authentication. The KDC maintains a database of the principals within its realm, their passwords, and other administrative information about each principal. A KDC can also accept authentication credentials from principals in other realms, which is known as a *cross-realm trust*. A common scenario for establishing a cross-realm trust relationship is to authenticate users from an Active Directory domain so that they can access an EMR cluster using their domain user account, using SSH to connect to the cluster, or working with big data applications and viewing Web interfaces.

Before you configure Kerberos using Amazon EMR, we recommend that you become familiar with Kerberos concepts, the services that run on a KDC, and the tools for administering Kerberos services. For more information, see [MIT Kerberos Documentation](#), which is published by the [Kerberos Consortium](#).

When you create a Kerberized cluster, Amazon EMR creates and configures a cluster-dedicated KDC that runs on the master node. Amazon EMR configures Kerberos for the applications, components, and subsystems that it installs on the cluster so that they are authenticated with each other. To set up users who authenticate using Kerberos, you can establish a cross-realm trust to authenticate users from a different KDC, or you can add users manually to the cluster-dedicated KDC. You can then configure Hadoop user directories so that they can use their Kerberos credentials to run jobs and connect to the cluster.

Topics

- [Supported Applications \(p. 144\)](#)
- [Configure Kerberos \(p. 145\)](#)
- [Configure a Cluster-Dedicated KDC \(p. 150\)](#)
- [Configure a Cross-Realm Trust \(p. 152\)](#)

Supported Applications

Within an EMR cluster, Kerberos principals are the big data application services and subsystems that run on all cluster nodes. Amazon EMR can configure the applications and components listed below to use Kerberos. Each application has a Kerberos user principal associated with it.

Amazon EMR does not support cross-realm trusts with AWS Directory Service for Microsoft Active Directory.

Amazon EMR only configures the open-source Kerberos authentication features for the applications and components listed below. Any other applications installed are not Kerberized, which can result in an inability to communicate with Kerberized components and cause application errors. Applications and components that are not Kerberized do not have authentication enabled. Supported applications and components may vary by Amazon EMR release version.

No web user interfaces hosted on the cluster are Kerberized.

- **HDFS**
- **YARN**
- **Tez**
- **Hadoop MapReduce**
- **Hbase**
- **HCatalog**
- **Hive**
 - Do not enable Hive with LDAP authentication. This may cause issues communicating with Kerberized YARN.
- **Hue**
 - Hue user authentication isn't set automatically and can be configured using the configuration API.
 - Hue server is Kerberized. The Hue front-end (UI) is not configured for authentication. LDAP authentication can be configured for the Hue UI.
- **Livy**
- **Oozie**
- **Spark**
- **Zeppelin**
 - Zeppelin is only configured to use Kerberos with the Spark interpreter. It is not configured for other interpreters.
 - Zeppelin impersonation with Kerberos is not supported. All users logged in to Zeppelin use the same Zeppelin user principal to run Spark jobs and authenticate to YARN.
- **Zookeeper**
 - Zookeeper client is not supported.

Configure Kerberos

Set up Kerberos on Amazon EMR by following these steps.

Step 1: Create a Security Configuration that Enables Kerberos and Optional Cross-Realm Trust Configuration

You can create a security configuration that specifies Kerberos attributes using the Amazon EMR console, the AWS CLI, or the EMR API. The security configuration can also contain other security options, such as encryption. For more information, see [Create a Security Configuration \(p. 175\)](#).

When you create a Kerberized cluster, you specify the security configuration together with Kerberos attributes that are specific to the cluster. You can't specify one set without the other, or an error occurs.

The following Kerberos parameters are set using the security configuration:

Parameter	Description
Enable Kerberos	Specifies that Kerberos is enabled for clusters that use this security configuration. If a cluster uses this security configuration, the cluster must also have Kerberos settings specified or an error occurs.
Ticket Lifetime	Specifies the period for which a Kerberos ticket issued by the cluster-dedicated KDC is valid. Ticket lifetimes are limited for security reasons. Cluster applications and services auto-renew tickets after they expire. Users who connect to the cluster over SSH using Kerberos credentials need to run <code>kinit</code> from the master node command line to renew after a ticket expires.
Cross-realm trust	If you provide a cross-realm trust configuration, principals (typically users) from another realm are authenticated to clusters that use this configuration. Additional configuration in the other Kerberos realm is also required. For more information, see Configure a Cross-Realm Trust (p. 152) .
Realm	Specifies the Kerberos realm name of the other realm in the trust relationship. Any string can be used, but by convention, this is typically the same as the Domain, but in all capital letters.
Domain	Specifies the domain name of the other realm in the trust relationship.
Admin server	Specifies the fully qualified domain name (FQDN) of the admin server in the other realm of the trust relationship. The admin server and KDC server typically run on the same machine with the same FQDN, but communicate on different ports. If no port is specified, port 749 is used, which is the Kerberos default. Optionally, you can specify the port (for example, <code>domain.example.com:749</code>).
KDC server	Specifies the fully qualified domain name (FQDN) of the KDC server in the other realm of the trust relationship. The KDC server and admin server typically run on the same machine with the same FQDN, but communicate on different ports. If no port is specified, port 88 is used, which is the Kerberos default. Optionally, you can specify the port (for example, <code>domain.example.com:88</code>).

The following examples demonstrate a security configuration with the same Kerberos attributes. The first example shows the security configuration created using the Amazon EMR console. The second example creates the same security configuration using a JSON file, which is referenced using the `create-security-configuration` command from the AWS CLI.

In the example, the KDC and admin services in the cross-realm trust are hosted on the same server, `ad.domain.com`. The default Kerberos ports of 749 for the KDC and 88 for administrative server are used. If your application uses customized ports, use the form `ad.domain.com:portnumber`.

Example Example Security Configuration with Kerberos Settings Using the Console

The screenshot shows the 'Create security configuration' page in the AWS Management Console. The left sidebar shows 'Clusters' and 'Security configurations' are selected. The main area has a 'Name' field set to 'KerberosSecurityConfig'. The 'Encryption' section is collapsed. The 'Authentication' section is expanded and highlighted with a red box. It contains a checked checkbox for 'Kerberos'. Below it, under 'Cross-realm trust', there are four input fields: 'Realm' (AD.DOMAIN.COM), 'Domain' (ad.domain.com), 'Admin server' (ad.domain.com), and 'KDC server' (ad.domain.com). Other sections like 'At-rest encryption', 'S3 encryption', 'Local disk encryption', and 'TLS certificate provider' are also present but not highlighted.

Example Example Security Configuration with Kerberos Settings Using the AWS CLI

The following `create-security-configuration` command creates a security configuration named `KerberosSecurityConfig` based on a JSON file, `MyKerberosSecurityConfig.json` saved locally in the same directory where you run the command.

```
aws emr create-security-configuration --name "KerberosSecurityConfiguration" --security-configuration file://MyKerberosSecurityConfig.json
```

The `MyKerberosSecurityConfig.json` uses the `AuthenticationConfiguration` object as shown below to specify Kerberos settings, which correspond to the console settings above. The security configuration is later referenced in the example `create-cluster` command [below \(p. 149\)](#).

```
{
  "AuthenticationConfiguration": {
    "KerberosConfiguration": {
```

```
        "Provider": "ClusterDedicatedKdc",
        "ClusterDedicatedKdcConfiguration": {
            "TicketLifetimeInHours": 24,
            "CrossRealmTrustConfiguration": {
                "Realm": "AD.DOMAIN.COM",
                "Domain": "ad.domain.com",
                "AdminServer": "ad.domain.com",
                "KdcServer": "ad.domain.com"
            }
        }
    }
}
```

Step 2: Configure Kerberos Attributes for a Cluster

Specify Kerberos attributes for a particular cluster along with the Kerberos security configuration when you create the cluster. You must specify the cluster Kerberos settings and a Kerberos security configuration together, or an error occurs. You can use the Amazon EMR console, the AWS CLI, or the Amazon EMR API.

The following Kerberos attributes are specified using the cluster configuration:

Attribute	Description
Realm	The Kerberos realm name for the cluster. The Kerberos convention is to set this to be the same as the domain name, but in uppercase. For example, for the domain <code>ec2.internal</code> , using <code>EC2.INTERNAL</code> as the realm name.
KDC admin password	The password used within the cluster for <code>kadmin</code> or <code>kadmin.local</code> . These are command-line interfaces to the Kerberos V5 administration system, which maintains Kerberos principals, password policies, and keytabs for the cluster.
Cross-realm trust principal password (optional)	Required when establishing a cross-realm trust. The cross-realm principal password, which must be identical across realms. Use a strong password.
AD domain join user (optional)	Required when establishing a cross-realm trust with an Active Directory domain. This is User logon name of an Active Directory account with sufficient privileges to join computers to the domain. Amazon EMR uses this identity to join the cluster to the domain. For more information, see the section called “Step 3: Add User Accounts to the Domain for the EMR Cluster” (p. 154) .
AD domain join password (optional)	The password for the Active Directory user that has sufficient privileges to join the cluster to the Active Directory domain. For more information, see the section called “Step 3: Add User Accounts to the Domain for the EMR Cluster” (p. 154) .

The following examples demonstrate the same configurations specified in the Amazon EMR console and using the `create-cluster` command from the AWS CLI.

Example Example Console Configuration

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

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

Security Options

EC2 key pair [Edit](#) [Help](#)
 Cluster visible to all IAM users in account [Help](#)

Permissions

Default Custom
Use default IAM roles. If roles are not present, they will be automatically created for you with managed policies for automatic policy updates.

EMR role [EMR_DefaultRole](#) [Edit](#) [Help](#)
EC2 instance profile [EMR_EC2_DefaultRole](#) [Edit](#) [Help](#)
Auto Scaling role [EMR_AutoScaling_DefaultRole](#) [Edit](#) [Help](#)

Authentication and encryption

Security configuration [KerberosSecurityConfig](#) [Edit](#) [Help](#)
Kerberos authentication is enabled in the KerberosSecurityConfig security configuration. Enter the appropriate values for your Kerberos configuration below.

Realm name EC2.INTERNAL
KDC admin password
Cross-realm trust password
AD joiner name ADUser
AD joiner password

[EC2 security groups](#)

[Cancel](#) [Previous](#) [Create cluster](#)

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 (^).

Example Example create-cluster Command

```
aws emr create-cluster --name "MyKerberosCluster" \
--release-label emr-5.10.0 \
--instance-type m4.large \
--instance-count 3 \
--use-default-roles \
--ec2-attributes KeyName=MyEC2KeyPair \
--security-configuration KerberosSecurityConfig \
--applications Name=Hadoop Name=Hive Name=Oozie Name=Hue Name=HCatalog Name=Spark \
--kerberos-attributes Realm=EC2.INTERNAL,KdcAdminPassword=MyVeryStrongPassword, \
CrossRealmTrustPrincipalPassword=MyVeryStrongMatchingPassword, \
ADDomainJoinUser=ADUser,ADDomainJoinPassword=MyADUserPassword
```

Step 3: Add Kerberos-Authenticated Users

Amazon EMR creates Kerberos-authenticated clients for applications that run on the cluster, for example, the Hadoop user, Spark user, and others. You can also add users who are authenticated to cluster processes using Kerberos. Authenticated users can connect to the cluster with their Kerberos credentials.

You can add users in either of the following ways:

- Configure a cross-realm trust to authenticate users from a different Kerberos realm, such as an Active Directory directory. For more information, see [Configure a Cross-Realm Trust \(p. 152\)](#).
- Add Linux accounts on the local cluster and add principals to the cluster-dedicated KDC for those accounts. For more information, see [Configure a Cluster-Dedicated KDC \(p. 150\)](#).

Important

The KDC, along with the database of principals, is lost when the master node terminates because the master node uses ephemeral storage. If you create users for SSH connections, we recommend that you establish a cross-realm trust with an external KDC configured for high-availability. Alternatively, if you create users for SSH connections using Linux user accounts, automate the account creation process using bootstrap actions and scripts so that it can be repeated when you create a new cluster.

Configure a Cluster-Dedicated KDC

You can set up your cluster without a cross-realm trust, manually adding Linux user accounts to all cluster nodes, adding Kerberos principals to the KDC on the master node, and ensuring that client computers have a Kerberos client installed.

Step 1: Create the Kerberized Cluster

1. Create a security configuration that enables Kerberos. The following example demonstrates a `create-security-configuration` command using the AWS CLI that specifies the security configuration as an inline JSON structure. You can also reference a file saved locally or in Amazon S3.

```
aws emr create-security-configuration --name MyKerberosConfig \  
--security-configuration '{"AuthenticationConfiguration": {"KerberosConfiguration": \  
{"Provider": "ClusterDedicatedKdc", "ClusterDedicatedKdcConfiguration": \  
{"TicketLifetimeInHours": 24}}}'
```

2. Create a cluster that references the security configuration, establishes Kerberos attributes for the cluster, and adds Linux accounts using a bootstrap action. The following example demonstrates a `create-cluster` command using the AWS CLI. The command references the security configuration that you created above, *MyKerberosConfig*. It also references a simple script, *createlinuxusers.sh*, as a bootstrap action, which you create and upload to Amazon S3 before creating the cluster.

```
aws emr create-cluster --name "MyKerberosCluster" \  
--release-label emr-5.10.0 \  
--instance-type m4.large \  
--instance-count 3 \  
--ec2-attributes InstanceProfile=EMR_EC2_DefaultRole,KeyName=MyEC2KeyPair \  
--service-role EMR_DefaultRole \  
--security-configuration MyKerberosConfig \  
--applications Name=Hadoop Name=Hive Name=Oozie Name=Hue Name=HCatalog Name=Spark \  
--kerberos-attributes Realm=EC2.INTERNAL, \  
KdcAdminPassword=MyClusterKDCAdminPwd \  
--bootstrap-actions Path=s3://mybucket/createlinuxusers.sh
```

The following example demonstrates the contents of the *createlinuxusers.sh* script, which adds user1, user2, and user3 to each node in the cluster. In the next step, you add these users as KDC principals.

```
#!/bin/bash  
sudo adduser user1  
sudo adduser user2  
sudo adduser user3
```

Step 2: Add Principals to the KDC, Create HDFS User Directories, and Configure SSH

The KDC running on the master node needs a principal added for the local host and for each user that you create on the cluster. You may also create HDFS directories for each user if they need to connect to the cluster and run Hadoop jobs. Similarly, configure the SSH service to enable GSSAPI authentication, which is required for Kerberos. After you enable GSSAPI, restart the SSH service.

The easiest way to accomplish these tasks is to submit a step to the cluster. The following example submits a bash script `configurekdc.sh` to the cluster you created in the previous step, referencing its cluster ID. The script is saved to Amazon S3. Alternatively, you could connect to the master node using an EC2 key pair to run the commands or submit the step during cluster creation.

```
aws emr add-steps --cluster-id j-01234567 --steps
  Type=CUSTOM_JAR,Name=CustomJAR,ActionOnFailure=CONTINUE,Jar=s3://
  myregion.elasticmapreduce/libs/script-runner/script-runner.jar,Args=[ "s3://mybucket/
  configurekdc.sh"]
```

The following example demonstrates the contents of the `configurekdc.sh` script.

```
#!/bin/bash
#Add a principal to the KDC for the master node, using the master node's returned host name
sudo kadmin.local -q "ktadd -k /etc/krb5.keytab host/`hostname -f`"
#Declare an associative array of user names and passwords to add
declare -A arr
arr=([user1]=pwd1 [user2]=pwd2 [user3]=pwd3)
for i in ${!arr[@]}; do
    #Assign plain language variables for clarity
    name=${i}
    password=${arr[$i]}

    # Create principal for sshuser in the master node and require a new password on first
    logon
    sudo kadmin.local -q "addprinc -pw $password +needchange $name"

    #Add user hdfs directory
    hdfs dfs -mkdir /user/$name

    #Change owner of user's hdfs directory to user
    hdfs dfs -chown $name:$name /user/$name
done

# Enable GSSAPI authentication for SSH and restart SSH service
sudo sed -i 's/^.*GSSAPIAuthentication.*$/GSSAPIAuthentication yes/' /etc/ssh/sshd_config
sudo sed -i 's/^.*GSSAPICleanupCredentials.*$/GSSAPICleanupCredentials yes/' /etc/ssh/
sshd_config
sudo /etc/init.d/sshd restart
```

Step 3: Connect Using SSH (Linux Client Computer)

Your Linux computer most likely includes an SSH client by default. For example, OpenSSH is installed on most Linux, Unix, and macOS operating systems. You can check for an SSH client by typing `ssh` at the command line. If your computer does not recognize the command, 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.

For more information about SSH connections, see [Connect to the Cluster \(p. 238\)](#). You must also have a Kerberos client installed.

The following procedure demonstrates the steps to connect to an EMR cluster using Kerberos from a Linux client using the [ssh command](#).

For [**MasterPublicDNS**](#), use the value that appears for **Master public DNS** on the **Summary** tab of the cluster details pane, for example, ec2-11-222-33-44.compute-1.amazonaws.com.

To use SSH to connect to a Kerberized EMR cluster from a Linux client

1. Use SSH to connect to the master node using an EC2 key pair and copy the contents of the /etc/krb5.conf file. For more information, see [Connect to the Cluster \(p. 238\)](#).
2. On each client computer that is used to connect to the cluster, create an identical /etc/krb5.conf file based on the copy that you made in the previous step.
3. Each time a user connects from a client computer, the user first renews Kerberos tickets as shown in the following example.

```
kinit user1
```

The user can then connect with ssh using a user name you created earlier and the public DNS name of the master node, as shown in the following example. Replace [`ec2-xx-xxx-xx-xx.compute-1.amazonaws.com`](#) with the **Master public DNS** value listed on the cluster's **Summary** page. The -K option specifies GSSAPI authentication.

```
ssh -K user1@ec2-xx-xxx-xx-xx.compute-1.amazonaws.com
```

Configure a Cross-Realm Trust

When you set up a cross-realm trust, you allow principals (usually users) from a different Kerberos realm to authenticate to application components on the EMR cluster. The cluster-dedicated KDC establishes a trust relationship with another KDC using a *cross-realm principal* that exists in both KDCs. The principal name and the password match precisely.

A cross-realm trust requires that the KDCs can reach each other over the network and resolve each other's domain names. Steps for establishing a cross-realm trust relationship with a Microsoft AD domain controller running as an EC2 instance are provided below, along with an example network setup that provides the required connectivity and domain-name resolution.

Important

Amazon EMR does not support cross-realm trusts with AWS Directory Service for Microsoft Active Directory.

Setting Up a Cross-Realm Trust with an Active Directory Domain Controller

[Step 1: Set Up the VPC and Subnet \(p. 153\)](#)

[Step 2: Launch and Install the Active Directory Domain Controller \(p. 154\)](#)

[Step 3: Add User Accounts to the Domain for the EMR Cluster \(p. 154\)](#)

[Step 4: Configure an Incoming Trust on the Active Directory Domain Controller \(p. 154\)](#)

[Step 5: Use a DHCP Option Set to Specify the Active Directory Domain Controller as a VPC DNS Server \(p. 154\)](#)

[Step 6: Launch a Kerberized EMR Cluster \(p. 155\)](#)

[Step 7: Create HDFS Users and Set Permissions on the Cluster for Active Directory User Accounts \(p. 155\)](#)

[Step 8: Use SSH to Log in to the Cluster \(p. 156\)](#)

Step 1: Set Up the VPC and Subnet

The following steps demonstrate creating a VPC and subnet so that the cluster-dedicated KDC can reach the Active Directory domain controller and resolve its domain name. In these steps, domain-name resolution is provided by referencing the Active Directory domain controller as the domain name server in the DHCP option set. For more information, see [Step 5: Use a DHCP Option Set to Specify the Active Directory Domain Controller as a VPC DNS Server \(p. 154\)](#).

The KDC and the Active Directory domain controller must be able to resolve each other's domain name. This allows Amazon EMR to join computers to the domain and automatically configure corresponding Linux user accounts and SSH parameters on cluster instances.

If Amazon EMR can't resolve the domain name, you can reference the trust using the Active Directory domain controller's IP address. However, you must manually add Linux user accounts, add corresponding principals to the cluster-dedicated KDC, and configure SSH.

To set up the VPC and subnet

1. Create an Amazon VPC with a single public subnet. For more information, see [Step 1: Create the VPC in the Amazon VPC Getting Started Guide](#).

Important

When you use a Microsoft Active Directory domain controller, choose a CIDR block for the EMR cluster so that all IPv4 addresses are fewer than nine characters in length (for example, 10.0.0.0/16). This is because the DNS names of cluster computers are used when the computers join the Active Directory directory. AWS assigns [DNS Hostnames](#) based on IPv4 address in a way that longer IP addresses may result in DNS names longer than 15 characters. Active Directory has a 15-character limit for registering joined computer names, and truncates longer names, which can cause unpredictable errors.

2. Remove the default DHCP option set assigned to the VPC. For more information, see [Changing a VPC to use No DHCP Options](#). Later on, you add a new one that specifies the Active Directory domain controller as the DNS server.
3. Confirm that DNS support is enabled for the VPC, that is, that DNS Hostnames and DNS Resolution are both enabled. They are enabled by default. For more information, see [Updating DNS Support for Your VPC](#).
4. Confirm that your VPC has an internet gateway attached, which is the default. For more information, see [Creating and Attaching an Internet Gateway](#).

Note

An internet gateway is used in this example because you are establishing a new domain controller for the VPC. An internet gateway may not be required for your application. The only requirement is that the cluster-dedicated KDC can access the Active Directory domain controller.

5. Create a custom route table, add a route that targets the Internet Gateway, and then attach it to your subnet. For more information, see [Create a Custom Route Table](#).
6. When you launch the EC2 instance for the domain controller, it must have a static public IPv4 address for you to connect to it using RDP. The easiest way to do this is to configure your subnet to auto-assign public IPv4 addresses. This is not the default setting when a subnet is created. For more information, see [Modifying the Public IPv4 Addressing Attribute of your Subnet](#). Optionally, you can assign the address when you launch the instance. For more information, see [Assigning a Public IPv4 Address During Instance Launch](#).
7. When you finish, make a note of your VPC and subnet IDs. You use them later when you launch the Active Directory domain controller and the cluster.

Step 2: Launch and Install the Active Directory Domain Controller

1. Launch an EC2 instance based on the Microsoft Windows Server 2016 Base AMI. We recommend an m4.xlarge or better instance type. For more information, see [Launching an AWS Marketplace Instance in the Amazon EC2 User Guide for Windows Instances](#).
2. Connect to the EC2 instance using RDP. For more information, see [Connecting to Your Windows Instance in the Amazon EC2 User Guide for Windows Instances](#).
3. Start **Server Manager** to install and configure the Active Directory Domain Services role on the server. Promote the server to a domain controller and assign a domain name (the example we use here is `ad.domain.com`). Make a note of the domain name because you need it later when you create the EMR security configuration and cluster. If you are new to setting up Active Directory, you can follow the instructions in [How to Set Up Active Directory \(AD\) in Windows Server 2016](#).

The instance restarts when you finish.

Step 3: Add User Accounts to the Domain for the EMR Cluster

RDP to the Active Directory domain controller to create user accounts in Active Directory Users and Computers for each cluster user. For instructions, see [Create a User Account in Active Directory Users and Computers](#). Make a note of each user's **User logon name**. You need these later when you configure the cluster.

In addition, create a user account with sufficient privileges to join computers to the domain. You specify this account when you create a cluster. Amazon EMR uses it to join cluster instances to the domain. You specify this account and its password in [Step 6: Launch a Kerberized EMR Cluster \(p. 155\)](#). To delegate computer join privileges to the user account, we recommend that you create a group with join privileges and then assign the user to the group. For instructions, see [Delegating Directory Join Privileges](#) in the [AWS Directory Service Administration Guide](#).

Step 4: Configure an Incoming Trust on the Active Directory Domain Controller

The example commands below create a trust in Active Directory, which is a one-way, incoming, non-transitive, realm trust with the cluster-dedicated KDC. The example we use for the cluster's realm is `EC2.INTERNAL`. The `passwordt` parameter specifies the **cross-realm principal password**, which you specify along with the cluster **realm** when you create a cluster. The realm name is derived from the default domain name in us-east-1 for the cluster. The **Domain** is the Active Directory domain in which you are creating the trust, which is lower case by convention. The example uses `ad.domain.com`

Open the Windows command prompt with administrator privileges and type the following commands to create the trust relationship on the Active Directory domain controller:

```
C:\Users\Administrator> ksetup /addkdc EC2.INTERNAL
C:\Users\Administrator> netdom trust EC2.INTERNAL /Domain:ad.domain.com /add /realm /
passwordt:MyVeryStrongPassword
C:\Users\Administrator> ksetup /SetEncTypeAttr EC2.INTERNAL AES256-CTS-HMAC-SHA1-96
```

Step 5: Use a DHCP Option Set to Specify the Active Directory Domain Controller as a VPC DNS Server

Now that the Active Directory domain controller is configured, you must configure the VPC to use it as a domain name server for name resolution within your VPC. To do this, attach a DHCP options set. Specify the **Domain name** as the domain name of your cluster—for example, `ec2.internal` if your cluster is in us-east-1 or `region.compute.amazonaws.com` for other regions. For **Domain name servers**, you must specify the IP address of the Active Directory domain controller (which must be reachable from the cluster) as the first entry, followed by **AmazonProvidedDNS** (for example, `xx.xx.xx.xx,AmazonProvidedDNS`). For more information, see [Changing DHCP Option Sets](#).

Step 6: Launch a Kerberized EMR Cluster

1. In Amazon EMR, create a security configuration that specifies the Active Directory domain controller you created in the previous steps. An example command is shown below. Replace the domain, `ad.domain.com`, with the name of the domain you specified in [Step 2: Launch and Install the Active Directory Domain Controller \(p. 154\)](#).

```
aws emr create-security-configuration --name MyKerberosConfig \
--security-configuration '{
    "AuthenticationConfiguration": {
        "KerberosConfiguration": {
            "Provider": "ClusterDedicatedKdc",
            "ClusterDedicatedKdcConfiguration": {
                "TicketLifetimeInHours": 24,
                "CrossRealmTrustConfiguration": {
                    "Realm": "AD.DOMAIN.COM",
                    "Domain": "ad.domain.com",
                    "AdminServer": "ad.domain.com",
                    "KdcServer": "ad.domain.com"
                }
            }
        }
    }
}'
```

2. Create the cluster, specifying the security configuration (in this example, `MyKerberosConfig`) and the same subnet you created in [Step 1: Set Up the VPC and Subnet \(p. 153\)](#).

Also specify the following cluster-specific `kerberos-attributes`:

- The realm for the cluster that you specified when you set up the Active Directory domain controller
- The cross-realm trust principal password that you specified as `passwordt` in [Step 4: Configure an Incoming Trust on the Active Directory Domain Controller \(p. 154\)](#).
- A `KdcAdminPassword`, which you can use to administer the cluster-dedicated KDC.
- The user logon name and password of the Active Directory account with computer join privileges that you created in [Step 3: Add User Accounts to the Domain for the EMR Cluster \(p. 154\)](#).

The following example launches a kerberized cluster.

```
aws emr create-cluster --name "MyKerberosCluster" \
--release-label emr-5.10.0 \
--instance-type m4.large \
--instance-count 3 \
--ec2-attributes InstanceProfile=EMR_EC2_DefaultRole,KeyName=MyEC2KeyPair \
--service-role EMR_DefaultRole \
--security-configuration MyKerberosConfig \
--applications Name=Hadoop Name=Hive Name=Oozie Name=Hue Name=HCatalog Name=Spark \
--kerberos-attributes Realm=EC2.INTERNAL, \
KdcAdminPassword=MyClusterKDCAdminPwd, \
ADDDomainJoinUser=ADUserLogonName, ADDDomainJoinPassword=ADUserPassword, \
CrossRealmTrustPrincipalPassword=MatchADTrustPwd
```

Step 7: Create HDFS Users and Set Permissions on the Cluster for Active Directory User Accounts

When setting up a trust relationship with Active Directory, Amazon EMR creates Linux users on the cluster for each Active Directory user account. For example, the user logon name LiJuan in Active Directory has a Linux user account of `lijuan`. Active Directory user names can contain upper-case letters, but Linux does not honor Active Directory casing.

To allow your users to log in to the cluster to run Hadoop jobs, you must add HDFS user directories for their Linux user accounts, and grant each user ownership of their directory. To do this, we recommend that you run a script saved to Amazon S3 as a cluster step. Alternatively, you can run the commands in the script below from the command line on the master node. Use the EC2 key pair that you specified when you created the cluster to connect to the master node over SSH as the Hadoop user. For more information, see [Use an Amazon EC2 Key Pair for SSH Credentials \(p. 157\)](#).

Run the following command to add a step to the cluster that runs a script, [AddHDFSUsers.sh](#).

```
aws emr add-steps --cluster-id ClusterID \
--steps Type=CUSTOM_JAR,Name=CustomJAR,ActionOnFailure=CONTINUE, \
Jar=s3://MyRegion.elasticmapreduce/libs/script-runner/script-runner.jar,Args=[ "s3:// \
MyBucketPath/AddHDFSUsers.sh" ]
```

The contents of the file [AddHDFSUsers.sh](#) is as follows.

```
#!/bin/bash
# AddHDFSUsers.sh script

# Initialize an array of user names from AD
ADUSERS=( "lijuan" "marymajor" "richardroe" "myusername" )

# For each user listed, create an HDFS user directory
# and change ownership to the user

for username in ${ADUSERS[@]}; do
    hdfs dfs -mkdir /user/$username
    hdfs dfs -chown $username:$username /user/$username
done
```

Active Directory Groups Mapped to Hadoop Groups

Amazon EMR uses System Security Services Daemon (SSD) to map Active Directory groups to Hadoop groups. To confirm group mappings, after you log in to the master node as described in the following step, you can use the `hdfs groups` command to confirm that Active Directory groups to which your Active Directory account belongs have been mapped to Hadoop groups for the corresponding Hadoop user on the cluster. You can also check other users' group mappings by specifying one or more user names with the command, for example `hdfs groups lijuan`. For more information, see [groups](#) in the [Apache HDFS Commands Guide](#).

Step 8: Use SSH to Log in to the Cluster

Users in the Active Directory domain should now be able to log on to the cluster with their domain credentials. Linux users can connect using `ssh` as shown in the following example. Replace `myusername` with the user logon name from Active Directory. Replace `ec2-xx-xxx-xx-xx.compute-1.amazonaws.com` with the **Master public DNS** value listed on the cluster's **Summary** page.

`myusername@ec2-xx-xxx-xx-xx.compute-1.amazonaws.com`

Your Linux computer most likely includes an SSH client by default. For example, OpenSSH is installed on most Linux, Unix, and macOS operating systems. You can check for an SSH client by typing `ssh` at the command line. If your computer does not recognize the command, 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.

Similarly, Windows users can use PuTTY, specifying `myusername@ec2-xx-xxx-xx-xx.compute-1.amazonaws.com` for the **Host Name**. Make sure that the default **Attempt GSSAPI Authentication** is still enabled under **Connection, SSH, Auth, GSSAPI**.

For more information about SSH connections, see [Connect to the Cluster \(p. 238\)](#).

Use an Amazon EC2 Key Pair for SSH Credentials

Amazon EMR can use an Amazon EC2 key pair to authorize SSH client connections to cluster instances. Usually, an SSH connection is used to connect directly to the master node to interact directly with system files and logs, or to view web interfaces that the master instance hosts. Alternatively, with Amazon EMR release version 5.10.0 or later, you can configure Kerberos to authenticate users and SSH connections to the master node. For more information, see [Use Kerberos Authentication \(p. 144\)](#).

You can use Amazon EC2 to create a key pair, or you can import a key pair. When you create a cluster, you specify the Amazon EC2 key pair to use for SSH connections. The SSH client that you use to connect needs the private key file associated with this key pair. This is a .pem file for SSH clients using Linux, Unix and macOS, and you must set permissions so that only the key owner has permission to access the file. This is a .ppk file for SSH clients using Windows, and the .ppk file is usually created from the .pem file.

- For more information about creating an Amazon EC2 key pair, see [Amazon EC2 Key Pairs](#) in the *Amazon EC2 User Guide for Linux Instances*.
- For instructions about using PuTTYgen to create a .ppk file from a .pem file, see [Converting Your Private Key Using PuTTYgen](#) in the *Amazon EC2 User Guide for Linux Instances*.
- For more information about setting .pem file permissions and how to connect to an EMR cluster's master node using different methods—including ssh from Linux or macOS, PuTTY from Windows, or the AWS CLI from any supported operating system, see [Connect to the Master Node Using SSH \(p. 239\)](#).

Encrypt Data in Transit and At Rest

Data encryption helps prevent unauthorized users from reading data on a cluster and associated data storage systems. This includes data saved to persistent media, known as data *at-rest*, and data that may be intercepted as it travels the network, known as data *in-transit*.

Beginning with Amazon EMR version 4.8.0, you can use Amazon EMR security configurations to configure data encryption settings for clusters more easily. Security configurations offer settings to enable security for data in-transit and data at-rest in Amazon Elastic Block Store (Amazon EBS) storage volumes and EMRFS on Amazon S3.

Optionally, beginning with Amazon EMR release version 4.1.0 and later, you can choose to configure transparent encryption in HDFS. For more information, see [Transparent Encryption in HDFS on Amazon EMR](#) in the *Amazon EMR Release Guide*. Beginning with Amazon EMR release version 5.7.0, you can specify a custom AMI with an encrypted EBS root device volume. For more information, see [Using a Custom AMI](#). These features are not configured using security configurations.

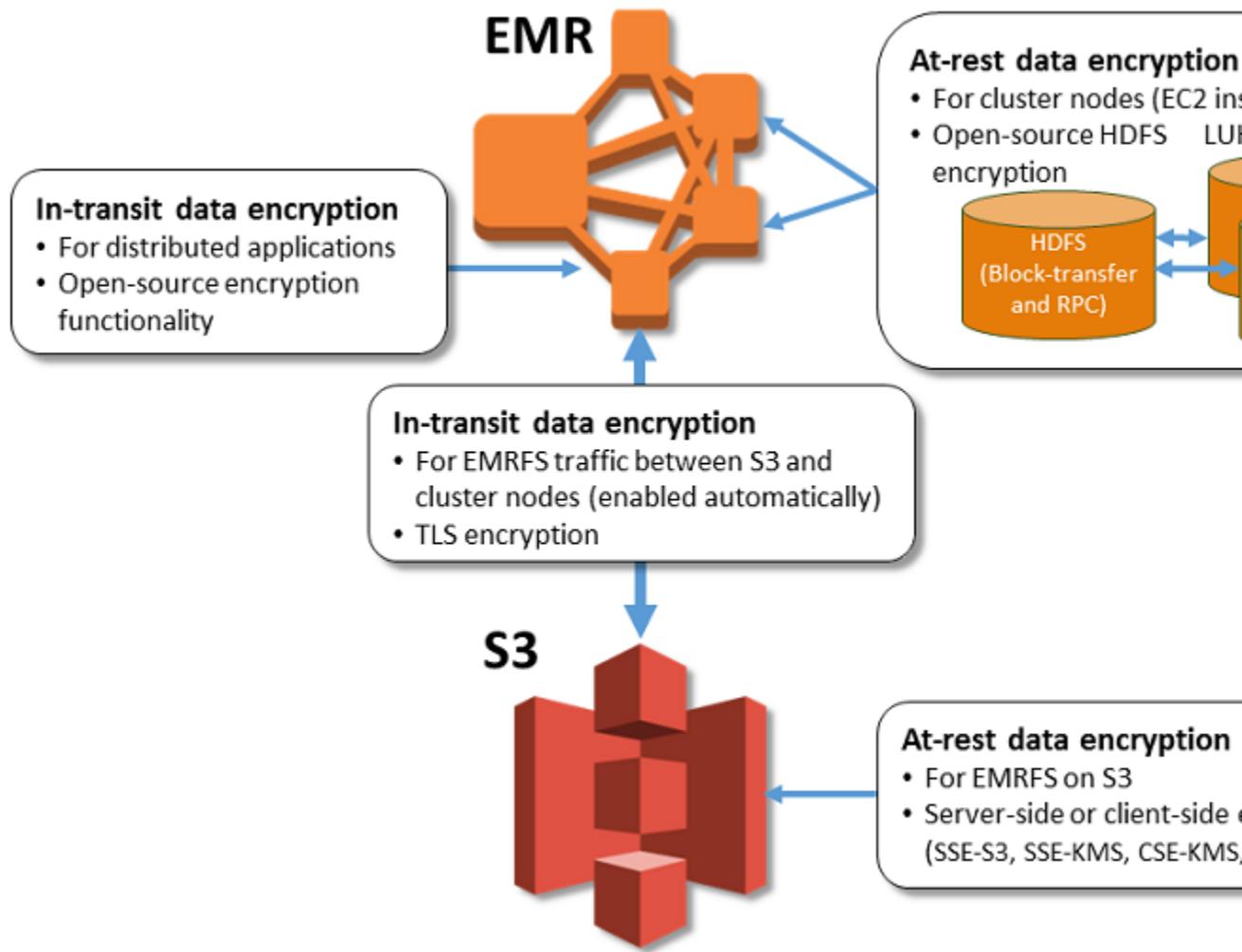
Topics

- [Encryption Options \(p. 157\)](#)
- [Create Keys and Certificates for Data Encryption \(p. 161\)](#)

Encryption Options

Beginning with Amazon EMR release version 4.8.0, you can use a security configuration to specify settings for encrypting data at-rest, data in-transit, or both. Each security configuration that you create is stored in Amazon EMR rather than in the cluster configuration, so you can easily reuse a configuration to specify data encryption settings whenever you create a cluster. For more information, see [Create a Security Configuration \(p. 175\)](#).

The following diagram shows the different data encryption options available with security configurations. When you enable at-rest data encryption, you specify options for encrypting EMRFS data in Amazon S3 and local disk encryption together. You can choose to enable only at-rest encryption, only in-transit encryption, or both.



The following encryption options are also available and are not configured using a security configuration:

- Using a security configuration does not encrypt the EBS root device volume on cluster instances. Beginning with Amazon EMR release version 5.7.0, you can do this by specifying a custom AMI with an encrypted EBS root device volume. For more information, see [Using a Custom AMI in the Amazon EMR Management Guide](#).
- If you are using a release version of Amazon EMR that does not support security configurations, you can configure encryption for EMRFS data in Amazon S3 manually. For more information, see [Specifying Amazon S3 Encryption Using EMRFS Properties \(p. 71\)](#).
- Optionally, beginning with Amazon EMR release version 4.1.0 and later, you can choose to configure transparent encryption in HDFS. For more information, see [Transparent Encryption in HDFS on Amazon EMR in the Amazon EMR Release Guide](#).

Data encryption requires keys and certificates. A security configuration gives you the flexibility to choose from several options, including keys managed by AWS Key Management Service, keys managed by Amazon S3, and keys and certificates from custom providers that you supply. When using AWS KMS as

your key provider, charges apply for the storage and use of encryption keys. For more information, see [AWS KMS Pricing](#).

Before you specify encryption options, decide on the key and certificate management systems you want to use, so you can first create the keys and certificates or the custom providers that you specify as part of encryption settings.

At-Rest Encryption for EMRFS data in Amazon S3

Amazon S3 encryption works with EMR File System (EMRFS) objects read from and written to Amazon S3. You specify Amazon S3 server-side encryption (SSE) or client-side encryption (CSE) when you enable at-rest encryption. Amazon S3 SSE and CSE encryption with EMRFS are mutually exclusive; you can choose either but not both. Regardless of whether Amazon S3 encryption is enabled, Transport Layer Security (TLS) encrypts the EMRFS objects in-transit between Amazon EMR cluster nodes and Amazon S3. For in-depth information about Amazon S3 encryption, see [Protecting Data Using Encryption](#) in the *Amazon Simple Storage Service Developer Guide*.

Amazon S3 Server-Side Encryption

When you set up Amazon S3 server-side encryption, Amazon S3 encrypts data at the object level as it writes the data to disk and decrypts the data when it is accessed. For more information about SSE, see [Protecting Data Using Server-Side Encryption](#) in the *Amazon Simple Storage Service Developer Guide*.

You can choose between two different key management systems when you specify SSE in Amazon EMR:

- **SSE-S3:** Amazon S3 manages keys for you.
- **SSE-KMS:** You use an AWS KMS customer master key (CMK) set up with policies suitable for Amazon EMR. For more information about key requirements for Amazon EMR, see [Using AWS KMS Customer Master Keys \(CMKs\) for Encryption \(p. 161\)](#). When you use AWS KMS, charges apply for the storage and use of encryption keys. For more information, see [AWS KMS Pricing](#).

SSE with customer-provided keys (SSE-C) is not available for use with Amazon EMR.

Amazon S3 Client-Side Encryption

With Amazon S3 client-side encryption, the Amazon S3 encryption and decryption takes place in the EMRFS client on your cluster. Objects are encrypted before being uploaded to Amazon S3 and decrypted after they are downloaded. The provider you specify supplies the encryption key that the client uses. The client can use keys provided by AWS KMS (CSE-KMS) or a custom Java class that provides the client-side master key (CSE-C). The encryption specifics are slightly different between CSE-KMS and CSE-C, depending on the specified provider and the metadata of the object being decrypted or encrypted. For more information about these differences, see [Protecting Data Using Client-Side Encryption](#) in the *Amazon Simple Storage Service Developer Guide*.

Note

Amazon S3 CSE only ensures that EMRFS data exchanged with Amazon S3 is encrypted; not all data on cluster instance volumes is encrypted. Furthermore, because Hue does not use EMRFS, objects that the Hue S3 File Browser writes to Amazon S3 are not encrypted.

At-rest Encryption for Local Disks

Local disk encryption within a security configuration applies to instance store and EBS storage volumes in a cluster. It does not apply to EBS root device volumes. Beginning with Amazon EMR version 5.7.0, you can specify a custom AMI to encrypt the EBS root device volumes of EC2 instances. This is a separate setting from security configurations. For more information, see [Using a Custom AMI](#) in the *Amazon EMR Management Guide*.

Two mechanisms work together to encrypt storage volumes when you enable at-rest data encryption:

- **Open-source HDFS Encryption:** HDFS exchanges data between cluster instances during distributed processing, and also reads from and writes data to instance store volumes and the EBS volumes attached to instances. The following open-source Hadoop encryption options are activated when you enable local-disk encryption:
 - [Secure Hadoop RPC](#) is set to "Privacy", which uses Simple Authentication Security Layer (SASL).
 - [Data encryption on HDFS block data transfer](#) is set to true and is configured to use AES 256 encryption.

Note

You can activate additional Apache Hadoop encryption by enabling in-transit encryption (see [In-Transit Data Encryption \(p. 160\)](#)). These encryption settings do not activate HDFS transparent encryption, which you can configure manually. For more information, see [Transparent Encryption in HDFS on Amazon EMR](#) in the *Amazon EMR Release Guide*.

- **LUKS.** In addition to HDFS encryption, the Amazon EC2 instance store volumes and the attached Amazon EBS volumes of cluster instances are encrypted using LUKS. For more information about LUKS encryption, see the [LUKS on-disk specification](#). At-rest encryption does not encrypt the EBS root device volume (boot volume). To encrypt the EBS root device volume, use Amazon EMR version 5.7.0 or later and specify a custom AMI. For more information, see [Customizing an AMI](#) in the *Amazon EMR Management Guide*.

For your key provider, you can use an AWS KMS CMK set up with policies suitable for Amazon EMR, or a custom Java class that provides the encryption artifacts. When you use AWS KMS, charges apply for the storage and use of encryption keys. For more information, see [AWS KMS Pricing](#).

In-Transit Data Encryption

Several encryption mechanisms are enabled with in-transit encryption. These are open-source features, are application-specific, and may vary by Amazon EMR release. The following application-specific encryption features can be enabled using security configurations:

- Hadoop (for more information, see [Hadoop in Secure Mode](#) in Apache Hadoop documentation):
 - [Hadoop MapReduce Encrypted Shuffle](#) uses TLS.
 - [Secure Hadoop RPC](#) is set to "Privacy" and uses SASL (activated in Amazon EMR when at-rest encryption is enabled).
 - [Data encryption on HDFS block data transfer](#) uses AES 256 (activated in Amazon EMR when at-rest encryption is enabled in the security configuration).
- HBase:
 - When Kerberos is enabled, the `hbase.rpc.protection` property is set to `privacy` for encrypted communication. For more information, see [Client-side Configuration for Secure Operation](#) in Apache HBase documentation. For more information about Kerberos with Amazon EMR, see [Use Kerberos Authentication \(p. 144\)](#).
- Presto:
 - Internal communication between Presto nodes uses SSL/TLS (Amazon EMR version 5.6.0 and later only).
- Tez:
 - [Tez Shuffle Handler](#) uses TLS (`tez.runtime.ssl.enable`).
- Spark (for more information, see [Spark security settings](#)):
 - Internal RPC communication between Spark components—for example, the block transfer service and the external shuffle service—is encrypted using the AES-256 cipher in Amazon EMR release version 5.9.0 and later. In earlier releases, internal RPC communication is encrypted using SASL with DIGEST-MD5 as the cipher.

- HTTP protocol communication with user interfaces such as Spark History Server and HTTPS-enabled file servers is encrypted using Spark's SSL configuration. For more information, see [SSL Configuration](#) in Spark documentation.

You specify the encryption artifacts used for in-transit encryption in one of two ways: either by providing a zipped file of certificates that you upload to Amazon S3, or by referencing a custom Java class that provides encryption artifacts. For more information, see [Providing Certificates for In-Transit Data Encryption with Amazon EMR Encryption \(p. 163\)](#).

Create Keys and Certificates for Data Encryption

Before you specify encryption options using a security configuration, decide on the provider you want to use for keys and encryption artifacts (for example, AWS KMS or a custom provider that you create) and create the keys or key provider as described in this section.

Providing Keys for At-Rest Data Encryption with Amazon EMR

You can use AWS Key Management Service (AWS KMS) or a custom key provider for at-rest data encryption in Amazon EMR. When you use AWS KMS, charges apply for the storage and use of encryption keys. For more information, see [AWS KMS Pricing](#).

This topic provides key policy details for an AWS KMS CMK to be used with Amazon EMR, as well as guidelines and code examples for writing a custom key provider class for Amazon S3 encryption. For more information about creating keys, see [Creating Keys](#) in the *AWS Key Management Service Developer Guide*.

Using AWS KMS Customer Master Keys (CMKs) for Encryption

The AWS KMS encryption key must be created in the same region as your Amazon EMR cluster instance and the Amazon S3 buckets used with EMRFS. If the key that you specify is in a different account from the one that you use to configure a cluster, you must specify the key using its ARN.

The role for the Amazon EC2 instance profile must have permissions to use the CMK you specify. The default role for the instance profile in Amazon EMR is `EMR_EC2_DefaultRole`. If you use a different role for the instance profile, or you use IAM roles for EMRFS requests to Amazon S3, make sure that each role is added as a key user as appropriate. This gives the role permissions to use the CMK. For more information, see [Using Key Policies](#) in the *AWS Key Management Service Developer Guide* and [Use Default IAM Roles and Managed Policies \(p. 188\)](#). Although you may use the same AWS KMS customer master key (CMK) for Amazon S3 data encryption as you use for local disk encryption, using separate keys is recommended.

You can use the AWS Management Console to add your instance profile or EC2 instance profile to the list of key users for the specified AWS KMS CMK, or you can use the AWS CLI or an AWS SDK to attach an appropriate key policy.

The procedure below describes how to add the default EMR instance profile, `EMR_EC2_DefaultRole` as a *key user* using the AWS Management Console. It assumes that you have already created a CMK. To create a new CMK, see [Creating Keys](#) in the *AWS Key Management Service Developer Guide*.

To add the EC2 instance profile for Amazon EMR to the list of encryption key users

1. Sign in to the AWS Management Console and open the AWS Key Management Service (AWS KMS) console at <https://console.aws.amazon.com/kms>.
2. To change the AWS Region, use the Region selector in the upper-right corner of the page.
3. Select the alias of the CMK to modify.

4. On the key details page under **Key Users**, choose **Add**.
5. In the **Attach** dialog box, select the appropriate role. The name of the default role is **EMR_EC2_DefaultRole**.
6. Choose **Attach**.

Creating a Custom Key Provider

When using a security configuration, you must specify a different provider class name for local disk encryption and Amazon S3 encryption.

When you create a custom key provider, the application is expected to implement the [EncryptionMaterialsProvider interface](#), which is available in the AWS SDK for Java version 1.11.0 and later. The implementation can use any strategy to provide encryption materials. You may, for example, choose to provide static encryption materials or integrate with a more complex key management system.

The encryption algorithm used for custom encryption materials must be **AES/GCM/NoPadding**.

The `EncryptionMaterialsProvider` class gets encryption materials by encryption context. Amazon EMR populates encryption context information at runtime to help the caller determine the correct encryption materials to return.

Example Example: Using a Custom Key Provider for Amazon S3 Encryption with EMRFS

When Amazon EMR fetches the encryption materials from the `EncryptionMaterialsProvider` class to perform encryption, EMRFS optionally populates the `materialsDescription` argument with two fields: the Amazon S3 URI for the object and the `JobFlowId` of the cluster, which can be used by the `EncryptionMaterialsProvider` class to return encryption materials selectively.

For example, the provider may return different keys for different Amazon S3 URI prefixes. It is the description of the returned encryption materials that is eventually stored with the Amazon S3 object rather than the `materialsDescription` value that is generated by EMRFS and passed to the provider. While decrypting an Amazon S3 object, the encryption materials description is passed to the `EncryptionMaterialsProvider` class, so that it can, again, selectively return the matching key to decrypt the object.

An `EncryptionMaterialsProvider` reference implementation is provided below. Another custom provider, [EMRFSRSAEncryptionMaterialsProvider](#), is available from GitHub.

```
import com.amazonaws.services.s3.model.EncryptionMaterials;
import com.amazonaws.services.s3.model.EncryptionMaterialsProvider;
import com.amazonaws.services.s3.model.KMSEncryptionMaterials;
import org.apache.hadoop.conf.Configurable;
import org.apache.hadoop.conf.Configuration;

import java.util.Map;

/**
 * Provides KMSEncryptionMaterials according to Configuration
 */
public class MyEncryptionMaterialsProviders implements EncryptionMaterialsProvider,
Configurable{
    private Configuration conf;
    private String kmsKeyId;
    private EncryptionMaterials encryptionMaterials;

    private void init() {
        this.kmsKeyId = conf.get("my.kms.key.id");
        this.encryptionMaterials = new KMSEncryptionMaterials(kmsKeyId);
```

```

    }

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

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

    @Override
    public void refresh() {

    }

    @Override
    public EncryptionMaterials getEncryptionMaterials(Map<String, String>
materialsDescription) {
        return this.encryptionMaterials;
    }

    @Override
    public EncryptionMaterials getEncryptionMaterials() {
        return this.encryptionMaterials;
    }
}

```

Providing Certificates for In-Transit Data Encryption with Amazon EMR Encryption

For in-transit data encryption using a security configuration (available in Amazon EMR release version 4.8.0 or later), you have two options to specify encryption artifacts:

- You can manually create PEM certificates, include them in a zip file, and then reference the zip file in Amazon S3.
- You can implement a custom certificate provider as a Java class. You specify the JAR file of the application in Amazon S3, and then provide the full class name of the provider as declared in the application. The class must implement the [TLSArtifactsProvider](#) interface available beginning with the AWS SDK for Java version 1.11.0.

Amazon EMR automatically downloads artifacts to each node in the cluster and later uses them to implement the open-source, in-transit encryption features. For more information about available options, see [In-Transit Data Encryption \(p. 160\)](#).

Using PEM Certificates

When you specify a zip file for in-transit encryption, the security configuration expects PEM files within the zip file to be named exactly as they appear below:

In-transit encryption certificates

File name	Required/optional	Details
privateKey.pem	Required	Private key
certificateChain.pem	Required	Certificate chain

File name	Required/optional	Details
trustedCertificates.pem	Optional	Required if the provided certificate is not signed by either the Java default trusted root certification authority (CA) or an intermediate CA that can link to the Java default trusted root CA. The Java default trusted root CAs can be found in <code>jre/lib/security/cacerts</code> .

You likely want to configure the private key PEM file to be a wildcard certificate that enables access to the Amazon VPC domain in which your cluster instances reside. For example, if your cluster resides in us-east-1 (N. Virginia), you could specify a common name in the certificate configuration that allows access to the cluster by specifying `CN=*.ec2.internal` in the certificate subject definition. If your cluster resides in us-west-2 (Oregon), you could specify `CN=*.us-west-2.compute.internal`. For more information about Amazon EMR cluster configuration within Amazon VPC, see [Select an Amazon VPC Subnet for the Cluster](#).

The following example demonstrates how to use [OpenSSL](#) to generate a self-signed X.509 certificate with a 1024-bit RSA private key. The key allows access to the issuer's Amazon EMR cluster instances in the us-west-2 (Oregon) region as specified by the `*.us-west-2.compute.internal` domain name as the common name.

Other optional subject items—such as country (C), state (S), Locale (L), etc.—are specified. Because a self-signed certificate is generated, the second command in the example copies the `certificateChain.pem` file to the `trustedCertificates.pem` file. The third command uses `zip` to create the `my-certs.zip` file that contains the certificates.

Important

This example is a proof-of-concept demonstration only. Using self-signed certificates is not recommended and presents a potential security risk. For production systems, use a trusted certification authority (CA) to issue certificates.

```
$ openssl req -x509 -newkey rsa:1024 -keyout privateKey.pem -out certificateChain.pem
  -days 365 -nodes -subj '/C=US/ST=Washington/L=Seattle/O=MyOrg/OU=MyDept/CN=*.us-
west-2.compute.internal'
$ cp certificateChain.pem trustedCertificates.pem
$ zip -r -X my-certs.zip certificateChain.pem privateKey.pem trustedCertificates.pem
```

Configure IAM Roles for EMRFS Requests to Amazon S3

If you have clusters with multiple users who need different levels of access to data in Amazon S3 through EMRFS, you can set up a security configuration to have EMRFS assume different IAM roles based on the user or group making the request, or based on the location of the data in Amazon S3. Each IAM role can have different permissions for data access in Amazon S3.

This feature is available with Amazon EMR release version 5.10.0 and later. If you use an earlier release version or have requirements beyond what IAM roles for EMRFS provide, you can create a custom credentials provider instead. For more information, see [Authorizing Access to EMRFS Data in Amazon S3 \(p. 70\)](#). For more information about EMRFS, see [Use EMR File System \(EMRFS\) \(p. 55\)](#).

How IAM Roles for EMRFS Work

By default, when a cluster application makes a request to Amazon S3 through EMRFS, EMRFS uses the permissions policies attached to the EMR role for EC2 that the cluster uses, regardless of the user or group using the application or the location of the data in Amazon S3.

When you use a security configuration to specify IAM roles for EMRFS, you set up role mappings. Each role mapping specifies an IAM role that corresponds to identifiers, which determine the basis for access to Amazon S3 through EMRFS. The identifiers can be users, groups, or Amazon S3 prefixes that indicate a data location. When EMRFS makes a request to Amazon S3 from a cluster that uses the security configuration, if the request matches the basis for access, EMRFS has cluster EC2 instances assume the corresponding IAM role for the request, and the IAM permissions attached to that role apply instead of the IAM permissions attached to the EMR role for EC2.

The users and groups in a role mapping are Hadoop users and groups that are defined on the cluster. Users and groups are passed to EMRFS in the context of the application using it (for example, YARN user impersonation). The Amazon S3 prefix can be a bucket specifier of any depth (for example, s3://mybucket or s3://mybucket/myproject/mydata). You can specify multiple identifiers within a single role mapping, but they all must be of the same type.

When a cluster application makes a request to Amazon S3 through EMRFS, EMRFS evaluates role mappings in the top-down order that they appear in the security configuration. If a request made through EMRFS doesn't match any identifier, EMRFS falls back to using the EMR role for EC2. For this reason, we recommend that the policies attached to this role limit permissions to Amazon S3.

Set Up a Security Configuration with IAM Roles for EMRFS

Before you set up a security configuration with IAM roles for EMRFS, plan and create the roles and permission policies to attach to the roles. For more information, see [How Do Roles for EC2 Instances Work?](#) in the *IAM User Guide*. When creating permissions policies, we recommend that you start with the managed policy attached to the default EMR role for EC2, which is `AmazonElasticMapReduceforEC2Role`, and edit this policy according to your requirements. For more information, see [Use Default IAM Roles and Managed Policies \(p. 188\)](#). If a role allows access to a location in Amazon S3 that is encrypted using an AWS Key Management Service customer master key (CMK), make sure that the role is specified as a key user. This gives the role permission to use the CMK. For more information, see [Using Key Policies](#) in the *AWS Key Management Service Developer Guide*.

Important

If none of the IAM roles for EMRFS that you specify apply, EMRFS falls back to the EMR role for EC2. Consider customizing this role to restrict permissions to Amazon S3 as appropriate for your application and then specifying this custom role instead of `EMR_EC2_DefaultRole` when you create a cluster. For more information, see [Customize IAM Roles \(p. 194\)](#) and [Specify Custom IAM Roles When You Create a Cluster \(p. 195\)](#).

To specify IAM roles for EMRFS requests to Amazon S3 using the console

1. Create a security configuration that specifies role mappings:
 - a. In the Amazon EMR console, select **Security configurations**, **Create**.
 - b. Type a **Name** for the security configuration. You use this name to specify the security configuration when you create a cluster.
 - c. Choose **Use IAM roles for EMRFS requests to Amazon S3**.
 - d. Select an **IAM role** to apply, and under **Basis for access** select an identifier type (**Users**, **Groups**, or **S3 prefixes**) from the list and enter corresponding identifiers. If you use multiple identifiers,

separate them with a comma and no space. For more information about each identifier type, see the [JSON configuration reference \(p. 166\)](#) below.

- e. Choose **Add role** to set up additional role mappings as described in the previous step.
- f. Set up other security configuration options as appropriate and choose **Create**. For more information, see [Create a Security Configuration \(p. 175\)](#).
2. Specify the security configuration you created above when you create a cluster. For more information, see [Specify a Security Configuration for a Cluster \(p. 186\)](#).

To specify IAM roles for EMRFS requests to Amazon S3 using the AWS CLI

1. Use the `aws emr create-security-configuration` command, specifying a name for the security configuration, and the security configuration details in JSON format.

The example command shown below creates a security configuration with the name `EMRFS_Roles_Security_Configuration`. It is based on a JSON structure in the file `MyEmrFsSecConfig.json`, which is saved in the same directory where the command is executed.

```
aws emr create-security-configuration --name EMRFS_Roles_Security_Configuration --  
security-configuration file://MyEmrFsSecConfig.json.
```

Use the following guidelines for the structure of the `MyEmrFsSecConfig.json` file. You can specify this structure along with structures for other security configuration options. For more information, see [Create a Security Configuration \(p. 175\)](#).

The following is an example JSON snippet for specifying custom IAM roles for EMRFS within a security configuration. It demonstrates role mappings for the three different identifier types, followed by a parameter reference.

```
{
  "AuthorizationConfiguration": {
    "EmrFsConfiguration": {
      "RoleMappings": [
        {
          "Role": "arn:aws:iam::123456789101:role/allow_EMRFS_access_for_user1",
          "IdentifierType": "User",
          "Identifiers": [ "user1" ]
        },
        {
          "Role": "arn:aws:iam::123456789101:role/allow_EMRFS_access_to_MyBuckets",
          "IdentifierType": "Prefix",
          "Identifiers": [ "s3://MyBucket/", "s3://MyOtherBucket/" ]
        },
        {
          "Role": "arn:aws:iam::123456789101:role/allow_EMRFS_access_for_AdminGroup",
          "IdentifierType": "Group",
          "Identifiers": [ "AdminGroup" ]
        }
      ]
    }
  }
}
```

Parameter	Description
<code>"AuthorizationConfiguration":</code>	Required.
<code>"EmrFsConfiguration":</code>	Required. Contains role mappings.
<code>"RoleMappings":</code>	Required. Contains one or more role mapping definitions. Role mappings are evaluated in the top-down order that they appear. If a role

Parameter	Description
	mapping evaluates as true for an EMRFS call for data in Amazon S3, no further role mappings are evaluated and EMRFS uses the specified IAM role for the request. Role mappings consist of the following required parameters:
"Role":	Specifies the ARN identifier of an IAM role in the format <code>arn:aws:iam::account-id:role/role-name</code> . This is the IAM role that Amazon EMR assumes if the EMRFS request to Amazon S3 matches any of the <code>Identifiers</code> specified.
"IdentifierType":	Can be one of the following: <ul style="list-style-type: none"> "User" specifies that the identifiers are one or more Hadoop users, which can be Linux account users or Kerberos principals. When the EMRFS request originates with the user or users specified, the IAM role is assumed. "Prefix" specifies that the identifier is an Amazon S3 location. The IAM role is assumed for calls to the location or locations with the specified prefixes. For example, the prefix <code>s3://mybucket/</code> matches <code>s3://mybucket/mydir</code> and <code>s3://mybucket/yetanotherdir</code>. "Group" specifies that the identifiers are one or more Hadoop groups. The IAM role is assumed if the request originates from a user in the specified group or groups.
"Identifiers":	Specifies one or more identifiers of the appropriate identifier type. Separate multiple identifiers by commas with no spaces.

2. Use the `aws emr create-cluster` command to create a cluster and specify the security configuration you created in the previous step.

The following example creates a cluster with default core Hadoop applications installed. The cluster uses the security configuration created above as `EMRFS_Roles_Security_Configuration` and also uses a custom EMR role for EC2, `EC2_Role_EMR_Restrict_S3`, which is specified using the `InstanceProfile` argument of the `--ec2-attributes` parameter.

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 MyEmrFsS3RolesCluster \
--release-label emr-5.20.0 --ec2-attributes
  InstanceProfile=EC2_Role_EMR_Restrict_S3,KeyName=MyKey \
--instance-type m4.large --instance-count 3 \
--security-configuration EMRFS_Roles_Security_Configuration
```

Control Network Traffic with Security Groups

Security groups act as virtual firewalls for EC2 instances in your cluster to control inbound and outbound traffic. Each security group has a set of rules that control inbound traffic, and a separate set of rules to control outbound traffic. For more information, see [Amazon EC2 Security Groups for Linux Instances](#).

You use two classes of security groups with Amazon EMR: *Amazon EMR-managed security groups* and *additional security groups*.

Every cluster has managed security groups associated with it. You can use the default managed security groups, or specify custom managed security groups. Either way, Amazon EMR automatically adds rules to managed security groups that a cluster needs to communicate between cluster instances and AWS services.

Additional security groups are optional. You can specify them in addition to managed security groups to tailor access to cluster instances. Additional security groups contain only rules that you define. Amazon EMR does not modify them.

The rules that Amazon EMR creates in managed security groups only allow the cluster to communicate among internal components. To allow users and applications to access a cluster from outside the cluster, you can edit rules in managed security groups, you can create additional security groups with additional rules, or do both.

For example, to allow a trusted client to connect to the master node using SSH, you must create an inbound rule associated with the master instance that allows SSH (Port 22) from the client. You can add this rule to the managed security group for the master instance after the cluster is created, or you can add it to an additional security group that you specify when you create the cluster.

You can specify security groups only when you create a cluster. They can't be added to a cluster or cluster instances while a cluster is running, but you can edit, add, and remove rules from existing security groups. The rules take effect as soon as you save them.

Security groups are restrictive by default. Unless a rule is added that allows traffic, the traffic is rejected. If there is more than one rule that applies to the same traffic and the same source, the most permissive rule applies. For example, if you have a rule that allows SSH from IP address 192.0.2.12/32, and another rule that allows access to all TCP traffic from the range 192.0.2.0/24, the rule that allows all TCP traffic from the range that includes 192.0.2.12 takes precedence. In this case, the client at 192.0.2.12 might have more access than you intended.

Important

Use caution when you edit security group rules. Be sure to add rules that only allow traffic from trusted clients for the protocols and ports that are required. We do not recommend any inbound rules that allow traffic from 0.0.0.0/0, which opens your cluster to that traffic from any client on the internet.

In addition, editing rules in managed security groups may have unintended consequences. You may inadvertently block the traffic required for clusters to function properly and cause errors because nodes are unreachable. Carefully plan and test security group configurations before implementation.

Topics

- [Working With Amazon EMR-Managed Security Groups \(p. 169\)](#)
- [Working With Additional Security Groups \(p. 172\)](#)
- [Specifying Amazon EMR-Managed and Additional Security Groups \(p. 173\)](#)

Working With Amazon EMR-Managed Security Groups

Different managed security groups are associated with the master instance and with the core and task instances in a cluster. An additional managed security group for service access is required when you create a cluster in a private subnet. For more information about the role of managed security groups with respect to your network configuration, see [Amazon VPC Options \(p. 103\)](#).

When you specify managed security groups for a cluster, you must use the same type of security group, default or custom, for all managed security groups. For example, you can't specify a custom security group for the master instance, and then not specify a custom security group for core and task instances.

If you use default managed security groups, you don't need to specify them when you create a cluster. Amazon EMR automatically uses the defaults. Moreover, if the defaults don't exist in the cluster's VPC yet, Amazon EMR creates them. Amazon EMR also creates them if you explicitly specify them and they don't exist yet.

You can edit rules in managed security groups after clusters are created. When you create a new cluster, Amazon EMR checks the rules in the managed security groups that you specify, and then creates any missing rules that the new cluster needs in addition to rules that may have been added earlier.

The default managed security groups are as follows:

- **ElasticMapReduce-master**

For rules in this security group, see [Amazon EMR-Managed Security Group for the Master Instance \(Public Subnets\) \(p. 169\)](#).

- **ElasticMapReduce-slave**

For rules in this security group, see [Amazon EMR-Managed Security Group for Core and Task Instances \(Public Subnets\) \(p. 170\)](#).

- **ElasticMapReduce-Master-Private**

For rules in this security group, see [Amazon EMR-Managed Security Group for the Master Instance \(Private Subnets\) \(p. 171\)](#).

- **ElasticMapReduce-Slave-Private**

For rules in this security group, see [Amazon EMR-Managed Security Group for Core and Task Instances \(Private Subnets\) \(p. 172\)](#).

- **ElasticMapReduce-ServiceAccess**

For rules in this security group, see [Amazon EMR-Managed Security Group for Service Access \(Private Subnets\) \(p. 172\)](#).

Amazon EMR-Managed Security Group for the Master Instance (Public Subnets)

The default managed security group for the master instance in public subnets has the **Group Name** of **ElasticMapReduce-master**. The default managed security group has the following rules, and Amazon EMR adds the same rules if you specify a custom managed security group.

Type	Protocol	Port Range	Source	Details
<i>Inbound rules</i>				
All ICMP-IPv4	All	N/A	The Group ID of the managed security group for the master instance. In other words, the same security group in which the rule appears.	These reflexive rules allow inbound traffic from any instance associated with the specified security group. Using the default <code>ElasticMapReduce-master</code> for multiple clusters allows the core and task nodes of those clusters to communicate with each other over ICMP or any TCP or UDP port. Specify custom managed security groups to restrict cross-cluster access.
All TCP	TCP	All		
All UDP	UDP	All		
All ICMP-IPv4	All	N/A	The Group ID of the managed security group specified for core and task nodes.	These rules allow all inbound ICMP traffic and traffic over any TCP or UDP port from any core and task instances that are associated with the specified security group, even if the instances are in different clusters.
All TCP	TCP	All		
All UDP	UDP	All		
Custom	TCP	8443	Various Amazon IP address ranges	These rules allow the cluster manager to communicate with the master node.
<i>Outbound rules</i>				
All traffic	All	All	0.0.0.0/0	This rule provides outbound access to the internet.

Amazon EMR-Managed Security Group for Core and Task Instances (Public Subnets)

The default managed security group for core and task instances in public subnets has the **Group Name** of `ElasticMapReduce-slave`. The default managed security group has the following rules, and Amazon EMR adds the same rules if you specify a custom managed security group.

Type	Protocol	Port Range	Source	Details
<i>Inbound rules</i>				
All ICMP-IPv4	All	N/A	The Group ID of the managed security group for core and task instances. In other words, the same security group in which the rule appears.	These reflexive rules allow inbound traffic from any instance associated with the specified security group. Using the default <code>ElasticMapReduce-slave</code> for multiple clusters allows the core and task instances of those clusters to communicate with each other over ICMP or any TCP or UDP port. Specify custom managed security groups to restrict cross-cluster access.
All TCP	TCP	All		
All UDP	UDP	All		
All ICMP-IPv4	All	N/A	The Group ID of the managed security group for the master instance.	These rules allow all inbound ICMP traffic and traffic over any TCP or UDP port from any master instances that are associated with the specified security group.
All TCP	TCP	All		

Type	Protocol	Port Range	Source	Details
All UDP	UDP	All		specified security group, even if the instances are in different clusters.
<i>Outbound rules</i>				
All traffic	All	All	0.0.0.0/0	Provides outbound access to the internet.

Amazon EMR-Managed Security Group for the Master Instance (Private Subnets)

The default managed security group for the master instance in private subnets has the **Group Name** of **ElasticMapReduce-Master-Private**. The default managed security group has the following rules, and Amazon EMR adds the same rules if you specify a custom managed security group.

Type	Protocol	Port Range	Source	Details
<i>Inbound rules</i>				
All ICMP-IPv4	All	N/A	The Group ID of the managed security group for the master instance. In other words, the same security group in which the rule appears.	These reflexive rules allow inbound traffic from any instance associated with the specified security group and reachable from within the private subnet. Using the default ElasticMapReduce-Master-Private for multiple clusters allows the core and task nodes of those clusters to communicate with each other over ICMP or any TCP or UDP port. Specify custom managed security groups to restrict cross-cluster access.
All TCP	TCP	All		
All UDP	UDP	All		
All ICMP-IPv4	All	N/A	The Group ID of the managed security group for core and task nodes.	These rules allow all inbound ICMP traffic and traffic over any TCP or UDP port from any core and task instances that are associated with the specified security group and reachable from within the private subnet, even if the instances are in different clusters.
All TCP	TCP	All		
All UDP	UDP	All		
HTTPS (8443)	TCP	8443	The Group ID of the managed security group for service access in a private subnet.	This rule allows the cluster manager to communicate with the master node.
<i>Outbound rules</i>				
All traffic	All	All	0.0.0.0/0	Provides outbound access to the internet.

Amazon EMR-Managed Security Group for Core and Task Instances (Private Subnets)

The default managed security group for core and task instances in private subnets has the **Group Name** of **ElasticMapReduce-Slave-Private**. The default managed security group has the following rules, and Amazon EMR adds the same rules if you specify a custom managed security group.

Type	Protocol	Port Range	Source	Details
<i>Inbound rules</i>				
All ICMP-IPV4	All	N/A	The Group ID of the managed security group for core and task instances. In other words, the same security group in which the rule appears.	These reflexive rules allow inbound traffic from any instance associated with the specified security group. Using the default <code>ElasticMapReduce-slave</code> for multiple clusters allows the core and task instances of those clusters to communicate with each other over ICMP or any TCP or UDP port. Specify custom managed security groups to restrict cross-cluster access.
All TCP	TCP	All		
All UDP	UDP	All		
All ICMP-IPV4	All	N/A	The Group ID of the managed security group for the master instance.	These rules allow all inbound ICMP traffic and traffic over any TCP or UDP port from any master instances that are associated with the specified security group, even if the instances are in different clusters.
All TCP	TCP	All		
All UDP	UDP	All		
HTTPS (8443)	TCP	8443	The Group ID of the managed security group for service access in a private subnet.	This rule allows the cluster manager to communicate with core and task nodes.
<i>Outbound rules</i>				
All traffic	All	All	0.0.0.0/0	Provides outbound access to the internet.

Amazon EMR-Managed Security Group for Service Access (Private Subnets)

The default managed security group for service access in private subnets has the **Group Name** of **ElasticMapReduce-ServiceAccess**. It has no inbound rules, and outbound rules that allow traffic over HTTPS (port 8443) to the other managed security groups in private subnets. This rule allows the cluster manager to communicate with core and task nodes. The same rules are added if you specify a custom security group.

Working With Additional Security Groups

Whether you use the default managed security groups or specify custom managed security groups, you can use additional security groups. Additional security groups give you the flexibility to tailor access between different clusters and from external clients, resources, and applications.

Consider the following scenario as an example. You have multiple clusters that you need to communicate with each other, but you want to allow inbound SSH access to the master instance for only a particular subset of clusters. To do this, you can use the same set of managed security groups for the clusters. You then create additional security groups that allow inbound SSH access from trusted clients, and specify the additional security groups for the master instance to each cluster in the subset.

You can apply up to four additional security groups for the master instance, four for core and task instances, and four for service access (in private subnets). If necessary, you can specify the same additional security group for master instances, core and task instances, and service access. The maximum number of security groups and rules in your account is subject to account limits. For more information, see [Security Group Limits](#) in the *Amazon VPC User Guide*.

Specifying Amazon EMR-Managed and Additional Security Groups

You can specify security groups using the AWS Management Console, the AWS CLI, or the EMR API. If you don't specify security groups, Amazon EMR creates default security groups. Specifying additional security groups is optional. You can assign additional security groups for master instances, core and task instances, and service access (private subnets only).

To specify security groups 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. Choose options for your cluster until you reach **Step 4: Security**.
4. Choose **EC2 Security Groups** to expand the section.

Under **EMR managed security groups**, the default managed security groups are selected by default. If a default doesn't exist in the VPC for **Master**, **Core & Task**, or **Service Access** (private subnet only), **Create** appears before the associated security group name.

5. If you use custom managed security groups, select them from the **EMR managed security groups** lists.

If you select a custom managed security group, a message notifies you to select a custom security group for other instances. You can use only custom or only default managed security groups for a cluster.

6. Optionally, under **Additional security groups**, choose the pencil icon, select up to four security groups from the list, and then choose **Assign security groups**. Repeat for each of **Master**, **Core & Task**, and **Service Access** as desired.
7. Choose **Create Cluster**.

Specifying Security Groups Using the AWS CLI

To specify security groups using the AWS CLI you use the `create-cluster` command with the following parameters of the `--ec2-attributes` option:

Parameter	Description
<code>EmrManagedMasterSecurityGroup</code>	Use this parameter to specify a custom managed security group for the master instance. If this parameter is specified, <code>EmrManagedSlaveSecurityGroup</code> you must also be specified. For clusters in private subnets,

Parameter	Description
	ServiceAccessSecurityGroup must also be specified.
EmrManagedSlaveSecurityGroup	Use this parameter to specify a custom managed security group for core and task instances. If this parameter is specified, EmrManagedMasterSecurityGroup you must also be specified. For clusters in private subnets, ServiceAccessSecurityGroup must also be specified.
ServiceAccessSecurityGroup	Use this parameter to specify a custom managed security group for service access, which applies only to clusters in private subnets. If this parameter is specified, EmrManagedMasterSecurityGroup and ServiceAccessSecurityGroup must also be specified.
AdditionalMasterSecurityGroups	Use this parameter to specify up to four additional security groups for the master instance.
AdditionalSlaveSecurityGroups	Use this parameter to specify up to four additional security groups for core and task instances.

Example — Specify Custom Amazon EMR-Managed Security Groups and Additional Security Groups

The following example specifies custom Amazon EMR managed security groups for a cluster in a private subnet, multiple additional security groups for the master instance, and a single additional security group for core and task instances.

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 "ClusterCustomManagedAndAdditionalSGs" \
--release-label emr-emr-5.20.0 --applications Name=Hue Name=Hive \
Name=Pig --use-default-roles --ec2-attributes \
SubnetIds=subnet-xxxxxxxxxxxx,KeyName=myKey, \
ServiceAccessSecurityGroup=sg-xxxxxxxxxxxx, \
EmrManagedMasterSecurityGroup=sg-xxxxxxxxxxxx, \
EmrManagedSlaveSecurityGroup=sg-xxxxxxxxxxxx, \
AdditionalMasterSecurityGroups=['sg-xxxxxxxxxxxx', \
'sg-xxxxxxxxxxxx', 'sg-xxxxxxxxxxxx'], \
AdditionalSlaveSecurityGroups=sg-xxxxxxxxxxxx \
--instance-type m4.large
```

For more information, see [create-cluster](#) in the *AWS CLI Command Reference*.

Use Security Configurations to Set Up Cluster Security

With Amazon EMR release version 4.8.0 or later, you can use security configurations to configure data encryption, Kerberos authentication (available in release version 5.10.0 and later), and Amazon S3 authorization for EMRFS (available in release version 5.10.0 or later).

After you create a security configuration, you specify it when you create a cluster, and you can re-use it for any number of clusters.

You can use the console, the AWS Command Line Interface (AWS CLI), or the AWS SDKs to create security configurations. You can also use an AWS CloudFormation template to create a security configuration. For more information, see [AWS CloudFormation User Guide](#) and the template reference for [AWS::EMR::SecurityConfiguration](#).

Topics

- [Create a Security Configuration \(p. 175\)](#)
- [Specify a Security Configuration for a Cluster \(p. 186\)](#)

Create a Security Configuration

This topic covers general procedures for creating a security configuration using the EMR console and the AWS CLI, followed by a reference for the parameters that comprise encryption, authentication, and IAM roles for EMRFS. For more information about these features, see the following topics:

- [Encrypt Data in Transit and At Rest \(p. 157\)](#)
- [Use Kerberos Authentication \(p. 144\)](#)
- [Configure IAM Roles for EMRFS Requests to Amazon S3 \(p. 164\)](#)

To create a security configuration using the console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. In the navigation pane, choose **Security Configurations**, **Create security configuration**.
3. Type a **Name** for the security configuration.
4. Choose options for **Encryption** and **Authentication** as described in the sections below and then choose **Create**.

To create a security configuration using the AWS CLI

- Use the `create-security-configuration` command as shown in the following example.
 - For `SecConfigName`, specify the name of the security configuration. This is the name you specify when you create a cluster that uses this security configuration.
 - For `SecConfigDef`, specify an inline JSON structure or the path to a local JSON file, such as `file://MySecConfig.json`. The JSON parameters define options for **Encryption**, **IAM Roles for EMRFS access to Amazon S3**, and **Authentication** as described in the sections below.

```
aws emr create-security-configuration --name "SecConfigName" --security-configuration SecConfigDef
```

Configure Data Encryption

Before you configure encryption in a security configuration, create the keys and certificates that are used for encryption. For more information, see [Providing Keys for At-Rest Data Encryption with Amazon EMR \(p. 161\)](#) and [Providing Certificates for In-Transit Data Encryption with Amazon EMR Encryption \(p. 163\)](#).

When you create a security configuration, you specify two sets of encryption options: at-rest data encryption and in-transit data encryption. Options for at-rest data encryption include both Amazon S3 with EMRFS and local-disk encryption. In-transit encryption options enable the open-source encryption features for certain applications that support Transport Layer Security (TLS). At-rest options and in-transit options can be enabled together or separately. For more information, see [Encrypt Data in Transit and At Rest \(p. 157\)](#).

Specifying Encryption Options Using the Console

Choose options under **Encryption** according to the following guidelines.

- Choose **At rest encryption** to encrypt data stored within the file system. This also enables Hadoop Distributed File System (HDFS) block-transfer encryption and RPC encryption, which need no further configuration.
- Under **S3 data encryption**, for **Encryption mode**, choose a value to determine how Amazon EMR encrypts Amazon S3 data with EMRFS.

What you do next depends on the encryption mode you chose:

- **SSE-S3**

Specifies [Server-side encryption with Amazon S3-managed encryption keys](#). You don't need to do anything more because Amazon S3 handles keys for you.

- **SSE-KMS or CSE-KMS**

Specifies [server-side encryption with AWS KMS-managed keys \(SSE-KMS\)](#) or [client-side encryption with AWS KMS-managed keys \(CSE-KMS\)](#). For **AWS KMS Key**, select a key. The key must exist in the same region as your Amazon EMR cluster. For key requirements, see [Using AWS KMS Customer Master Keys \(CMKs\) for Encryption \(p. 161\)](#).

- **CSE-Custom**

Specifies [client-side encryption using a custom client-side master key \(CSE-Custom\)](#). For **S3 object**, enter the location in Amazon S3, or the Amazon S3 ARN, of your custom key-provider JAR file. Then, for **Key provider class**, enter the full class name of a class declared in your application that implements the `EncryptionMaterialsProvider` interface.

- Under **Local disk encryption**, choose a value for **Key provider type**. Amazon EMR uses this key for Linux Unified Key System (LUKS) encryption for the local volumes (except boot volumes) attached to your cluster nodes.

- **AWS KMS**

Select this option to specify an AWS KMS customer master key (CMK). For **AWS KMS Key**, select a key. The key must exist in the same region as your Amazon EMR cluster. For more information about key requirements, see [Using AWS KMS Customer Master Keys \(CMKs\) for Encryption \(p. 161\)](#).

- **Custom**

Select this option to specify a custom key provider. For **S3 object**, enter the location in Amazon S3, or the Amazon S3 ARN, of your custom key-provider JAR file. For **Key provider class**, enter the full class name of a class declared in your application that implements the `EncryptionMaterialsProvider` interface. The class name you provide here must be different from the class name provided for CSE-Custom.

- Choose **In-transit encryption** to enable the open-source TLS encryption features for in-transit data. Choose a **Certificate provider type** according to the following guidelines:

- **PEM**

Select this option to use PEM files that you provide within a zip file. Two artifacts are required within the zip file: privateKey.pem and certificateChain.pem. A third file, trustedCertificates.pem, is optional. See [Providing Certificates for In-Transit Data Encryption with Amazon EMR Encryption \(p. 163\)](#) for details. For **S3 object**, specify the location in Amazon S3, or the Amazon S3 ARN, of the zip file field.

- **Custom**

Select this option to specify a custom certificate provider and then, for **S3 object**, enter the location in Amazon S3, or the Amazon S3 ARN, of your custom certificate-provider JAR file. For **Key provider class**, enter the full class name of a class declared in your application that implements the TLSArtifactsProvider interface.

Specifying Encryption Options Using the AWS CLI

The sections that follow use sample scenarios to illustrate well-formed **--security-configuration** JSON for different configurations and key providers, followed by a reference for the JSON parameters and appropriate values.

Example In-Transit Data Encryption Options

The example below illustrates the following scenario:

- In-transit data encryption is enabled and at-rest data encryption is disabled.
- A zip file with certificates in Amazon S3 is used as the key provider (see [Providing Certificates for In-Transit Data Encryption with Amazon EMR Encryption \(p. 163\)](#) for certificate requirements).

```
aws emr create-security-configuration --name "MySecConfig" --security-configuration '{  
    "EncryptionConfiguration": {  
        "EnableInTransitEncryption" : true,  
        "EnableAtRestEncryption" : false,  
        "InTransitEncryptionConfiguration" : {  
            "TLSCertificateConfiguration" : {  
                "CertificateProviderType" : "PEM",  
                "S3Object" : "s3://MyConfigStore/artifacts/MyCerts.zip"  
            }  
        }  
    }  
}'
```

The example below illustrates the following scenario:

- In-transit data encryption is enabled and at-rest data encryption is disabled.
- A custom key provider is used (see [Providing Certificates for In-Transit Data Encryption with Amazon EMR Encryption \(p. 163\)](#) for certificate requirements).

```
aws emr create-security-configuration --name "MySecConfig" --security-configuration '{  
    "EncryptionConfiguration": {  
        "EnableInTransitEncryption" : true,  
        "EnableAtRestEncryption" : false,  
    }  
}'
```

```

    "InTransitEncryptionConfiguration" : {
      "TLSCertificateConfiguration" : {
        "CertificateProviderType" : "Custom",
        "S3Object" : "s3://MyConfig/artifacts/MyCerts.jar",
        "CertificateProviderClass" : "com.mycompany.MyCertProvider"
      }
    }
  }
}

```

Example At-Rest Data Encryption Options

The example below illustrates the following scenario:

- In-transit data encryption is disabled and at-rest data encryption is enabled
- SSE-S3 is used for Amazon S3 encryption
- Local disk encryption uses AWS KMS as the key provider

```

aws emr create-security-configuration --name "MySecConfig" --security-configuration '{
  "EncryptionConfiguration": {
    "EnableInTransitEncryption" : false,
    "EnableAtRestEncryption" : true,
    "AtRestEncryptionConfiguration" : {
      "S3EncryptionConfiguration" : {
        "EncryptionMode" : "SSE-S3"
      },
      "LocalDiskEncryptionConfiguration" : {
        "EncryptionKeyProviderType" : "AwsKms",
        "AwsKmsKey" : "arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012"
      }
    }
  }
}'

```

The example below illustrates the following scenario:

- In-transit data encryption is enabled and references a zip file with PEM certificates in Amazon S3, using the ARN
- SSE-KMS is used for Amazon S3 encryption
- Local disk encryption uses AWS KMS as the key provider

```

aws emr create-security-configuration --name "MySecConfig" --security-configuration '{
  "EncryptionConfiguration": {
    "EnableInTransitEncryption" : true,
    "EnableAtRestEncryption" : true,
    "InTransitEncryptionConfiguration" : {
      "TLSCertificateConfiguration" : {
        "CertificateProviderType" : "PEM",
        "S3Object" : "arn:aws:s3:::MyConfigStore/artifacts/MyCerts.zip"
      }
    },
    "AtRestEncryptionConfiguration" : {
      "S3EncryptionConfiguration" : {
        "EncryptionMode" : "SSE-KMS",
        "AwsKmsKey" : "arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012"
      }
    }
}'

```

```

    "LocalDiskEncryptionConfiguration" : {
        "EncryptionKeyProviderType" : "AwsKms",
        "AwsKmsKey" : "arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012"
    }
}
}
}'

```

The example below illustrates the following scenario:

- In-transit data encryption is enabled and references a zip file with PEM certificates in Amazon S3
- CSE-KMS is used for Amazon S3 encryption
- Local disk encryption uses a custom key provider referenced by its ARN

```

aws emr create-security-configuration --name "MySecConfig" --security-configuration '{
    "EncryptionConfiguration": {
        "EnableInTransitEncryption" : true,
        "EnableAtRestEncryption" : true,
        "InTransitEncryptionConfiguration" : {
            "TLSCertificateConfiguration" : {
                "CertificateProviderType" : "PEM",
                "S3Object" : "s3://MyConfigStore/artifacts/MyCerts.zip"
            }
        },
        "AtRestEncryptionConfiguration" : {
            "S3EncryptionConfiguration" : {
                "EncryptionMode" : "CSE-KMS",
                "AwsKmsKey" : "arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012"
            }
        },
        "LocalDiskEncryptionConfiguration" : {
            "EncryptionKeyProviderType" : "Custom",
            "S3Object" : "arn:aws:s3:::artifacts/MyKeyProvider.jar",
            "EncryptionKeyProviderClass" : "com.mycompany.MyKeyProvider.jar"
        }
    }
}'

```

The example below illustrates the following scenario:

- In-transit data encryption is enabled with a custom key provider
- CSE-Custom is used for Amazon S3 data
- Local disk encryption uses a custom key provider

```

aws emr create-security-configuration --name "MySecConfig" --security-configuration '{
    "EncryptionConfiguration": {
        "EnableInTransitEncryption" : "true",
        "EnableAtRestEncryption" : "true",
        "InTransitEncryptionConfiguration" : {
            "TLSCertificateConfiguration" : {
                "CertificateProviderType" : "Custom",
                "S3Object" : "s3://MyConfig/artifacts/MyCerts.jar",
                "CertificateProviderClass" : "com.mycompany.MyCertProvider"
            }
        },
    }
}'

```

```

    "AtRestEncryptionConfiguration" : {
      "S3EncryptionConfiguration" : {
        "EncryptionMode" : "CSE-Custom",
        "S3Object" : "s3://MyConfig/artifacts/MyCerts.jar",
        "EncryptionKeyProviderClass" : "com.mycompany.MyKeyProvider"
      },
      "LocalDiskEncryptionConfiguration" : {
        "EncryptionKeyProviderType" : "Custom",
        "S3Object" : "s3://MyConfig/artifacts/MyCerts.jar",
        "EncryptionKeyProviderClass" : "com.mycompany.MyKeyProvider"
      }
    }
  }
}

```

JSON Reference for Encryption Settings

The following table lists the JSON parameters for encryption settings and provides a description of acceptable values for each parameter.

Parameter	Description
"EnableInTransitEncryption" : true false	Specify true to enable in-transit encryption and false to disable it. If omitted, false is assumed, and in-transit encryption is disabled.
"EnableAtRestEncryption" : true false	Specify true to enable at-rest encryption and false to disable it. If omitted, false is assumed and at-rest encryption is disabled.
In-transit encryption parameters	
"InTransitEncryptionConfiguration" :	Specifies a collection of values used to configure in-transit encryption when EnableInTransitEncryption is true.
"CertificateProviderType" : "PEM" "Custom"	Specifies whether to use PEM certificates referenced with a zipped file, or a Custom certificate provider. If PEM is specified, S3Object must be a reference to the location in Amazon S3 of a zip file containing the certificates. If Custom is specified, S3Object must be a reference to the location in Amazon S3 of a JAR file, followed by a CertificateProviderClass entry.
"S3Object" : " <i>zipLocation</i> " " <i>JarLocation</i> "	Provides the location in Amazon S3 to a zip file when PEM is specified, or to a JAR file when Custom is specified. The format can be a path (for example, s3://MyConfig/artifacts/CertFiles.zip) or an ARN (for example, arn:aws:s3:::Code/MyCertProvider.jar). If a zip file is specified, it must contain files named exactly privateKey.pem and certificateChain.pem. A file named trustedCertificates.pem is optional.
"CertificateProviderClass" : " <i>MyClassID</i> "	Required only if Custom is specified for CertificateProviderType. <i>MyClassID</i> specifies a full class name declared in the JAR file, which implements the

Parameter	Description
	TLSArtifactsProvider interface. For example, <code>com.mycompany.MyCertProvider</code> .
At-rest encryption parameters	
"AtRestEncryptionConfiguration" :	Specifies a collection of values for at-rest encryption when <code>EnableAtRestEncryption</code> is true, including Amazon S3 encryption and local disk encryption.
Amazon S3 encryption parameters	
"S3EncryptionConfiguration" :	Specifies a collection of values used for Amazon S3 encryption with the EMR File System (EMRFS).
"EncryptionMode" : "SSE-S3" "SSE-KMS" "CSE-KMS" "CSE-Custom"	Specifies the type of Amazon S3 encryption to use. If SSE-S3 is specified, no further Amazon S3 encryption values are required. If either SSE-KMS or CSE-KMS is specified, an AWS KMS customer master key (CMK) ARN must be specified as the <code>AwsKmsKey</code> value. If CSE-Custom is specified, <code>S3Object</code> and <code>EncryptionKeyProviderClass</code> values must be specified.
"AwsKmsKey" : " <i>MyKeyARN</i> "	Required only when either SSE-KMS or CSE-KMS is specified for <code>EncryptionMode</code> . <i>MyKeyARN</i> must be a fully specified ARN to a key (for example, <code>arn:aws:kms:us-east-1:123456789012:key/12345678-1234-1</code>).
"S3Object" : " <i>JarLocation</i> "	Required only when CSE-Custom is specified for <code>CertificateProviderType</code> . <i>JarLocation</i> provides the location in Amazon S3 to a JAR file. The format can be a path (for example, <code>s3://MyConfig/artifacts/MyKeyProvider.jar</code>) or an ARN (for example, <code>arn:aws:s3:::Code/MyKeyProvider.jar</code>).
"EncryptionKeyProviderClass" : " <i>MyS3KeyClassID</i> "	Required only when CSE-Custom is specified for <code>EncryptionMode</code> . <i>MyS3KeyClassID</i> specifies a full class name of a class declared in the application that implements the <code>EncryptionMaterialsProvider</code> interface; for example, <code>com.mycompany.MyS3KeyProvider</code> .
Local disk encryption parameters	
"LocalDiskEncryptionKeyProvider"	Specifies the key provider and corresponding values to be used for local disk encryption.
"Type" : "AwsKms" "Custom"	Specifies the key provider. If <code>AwsKms</code> is specified, an AWS KMS CMK ARN must be specified as the <code>AwsKmsKey</code> value. If <code>Custom</code> is specified, <code>S3Object</code> and <code>EncryptionKeyProviderClass</code> values must be specified.

Parameter	Description
"AwsKmsKey" : " <i>MyKeyARN</i> "	Required only when <code>AwsKms</code> is specified for <code>Type</code> . <i>MyKeyARN</i> must be a fully specified ARN to a key (for example, <code>arn:aws:kms:us-east-1:123456789012:key/12345678-1234-1234-1234-1234-1234-1234-1234</code>).
"S3Object" : " <i>JarLocation</i> "	Required only when <code>CSE-Custom</code> is specified for <code>CertificateProviderType</code> . <i>JarLocation</i> provides the location in Amazon S3 to a JAR file. The format can be a path (for example, <code>s3://MyConfig/artifacts/MyKeyProvider.jar</code>) or an ARN (for example, <code>arn:aws:s3:::Code/MyKeyProvider.jar</code>).
"EncryptionKeyProviderClass" : " <i>MyLocalDiskKeyClassID</i> "	Required only when <code>Custom</code> is specified for <code>Type</code> . <i>MyLocalDiskKeyClassID</i> specifies a full class name of a class declared in the application that implements the <code>EncryptionMaterialsProvider</code> interface; for example, <code>com.mycompany.MyLocalDiskKeyProvider</code> .

Configure Kerberos Authentication

A security configuration with Kerberos settings can only be used by a cluster that is created with Kerberos attributes or an error occurs. For more information, see [Configure Kerberos \(p. 145\)](#). Kerberos is only available in Amazon EMR release version 5.10.0 and later.

Specifying Kerberos Settings Using the Console

Choose options under **Kerberos authentication** according to the following guidelines.

Parameter	Description
Enable Kerberos	Specifies that Kerberos is enabled for clusters that use this security configuration. If a cluster uses this security configuration, the cluster must also have Kerberos settings specified or an error occurs.
Ticket Lifetime	Specifies the period for which a Kerberos ticket issued by the cluster-dedicated KDC is valid. Ticket lifetimes are limited for security reasons. Cluster applications and services auto-renew tickets after they expire. Users who connect to the cluster over SSH using Kerberos credentials need to run <code>kinit</code> from the master node command line to renew after a ticket expires.
Cross-realm trust	If you provide a cross-realm trust configuration, principals (typically users) from another realm are authenticated to clusters that use this configuration. Additional configuration in the other Kerberos realm is also required. For more information, see Configure a Cross-Realm Trust (p. 152) .

Parameter	Description
Realm	Specifies the Kerberos realm name of the other realm in the trust relationship. Any string can be used, but by convention, this is typically the same as the Domain, but in all capital letters.
Domain	Specifies the domain name of the other realm in the trust relationship.
Admin server	Specifies the fully qualified domain name (FQDN) of the admin server in the other realm of the trust relationship. The admin server and KDC server typically run on the same machine with the same FQDN, but communicate on different ports. If no port is specified, port 749 is used, which is the Kerberos default. Optionally, you can specify the port (for example, domain.example.com:749).
KDC server	Specifies the fully qualified domain name (FQDN) of the KDC server in the other realm of the trust relationship. The KDC server and admin server typically run on the same machine with the same FQDN, but communicate on different ports. If no port is specified, port 88 is used, which is the Kerberos default. Optionally, you can specify the port (for example, domain.example.com:88).

Specifying Kerberos Settings Using the AWS CLI

The following example shows JSON parameters for a Kerberos configuration, followed by a reference for parameters and values.

```
{
    "AuthenticationConfiguration": {
        "KerberosConfiguration": {
            "Provider": "ClusterDedicatedKdc",
            "ClusterDedicatedKdcConfiguration": {
                "TicketLifetimeInHours": number,
                "CrossRealmTrustConfiguration": {
                    "Realm": "DOMAIN.EXAMPLE.COM",
                    "Domain": "domain.example.com",
                    "AdminServer": "domain.example.com",
                    "KdcServer": "domain.example.com"
                }
            }
        }
    }
}
```

Parameter	Description
<code>"AuthenticationConfiguration" :</code>	Required. Contains Kerberos configuration parameters.
<code>"KerberosConfiguration" :</code>	Required. Contains Kerberos configuration parameters.

Parameter	Description
"Provider": "ClusterDedicatedKdc"	Required. Specifies that a cluster-dedicated KDC is created on the master node.
"ClusterDedicatedKdcConfiguration"	Required. Contains configuration parameters for the cluster-dedicated KDC on the master node.
"TicketLifetimeInHours": <i>number</i>	Optional. If omitted, defaults to 24. Specifies the period for which a Kerberos ticket issued by the cluster-dedicated KDC is valid. Ticket lifetimes are limited for security reasons. Cluster applications and services auto-renew tickets after they expire. Users who connect to the cluster over SSH using Kerberos credentials need to run <code>kinit</code> from the master node command line to renew after a ticket expires.
"CrossRealmTrustConfiguration":	Optional. Contains parameters that define a cross-realm trust configuration. If you provide a cross-realm trust configuration, principals (typically users) from another realm are authenticated to clusters that use this configuration. Additional configuration in the other Kerberos realm is also required. For more information, see Configure a Cross-Realm Trust (p. 152) .
"Realm": " <i>DOMAIN.EXAMPLE.COM</i> "	Specifies the Kerberos realm name of the other realm in the trust relationship. Any string can be used, but by convention, this is typically the same as the Domain, but in all capital letters.
"Domain": " <i>domain.example.com</i> "	Specifies the domain name of the other realm in the trust relationship.
"AdminServer": " <i>domain.example.com</i> "	Specifies the fully qualified domain name (FQDN) of the admin server in the other realm of the trust relationship. The admin server and KDC server typically run on the same machine with the same FQDN, but communicate on different ports. If no port is specified, port 749 is used, which is the Kerberos default. Optionally, you can specify the port (for example, <code>domain.example.com:749</code>).
"KdcServer": " <i>domain.example.com</i> "	Specifies the fully qualified domain name (FQDN) of the KDC server in the other realm of the trust relationship. The KDC server and admin server typically run on the same machine with the same FQDN, but communicate on different ports. If no port is specified, port 88 is used, which is the Kerberos default. Optionally, you can specify the port (for example, <code>domain.example.com:88</code>).

Configure IAM Roles for EMRFS Requests to Amazon S3

IAM roles for EMRFS allow you to provide different permissions to EMRFS data in Amazon S3. You create mappings that specify an IAM role that is used for permissions when an access request contains an identifier that you specify. The identifier can be a Hadoop user or role, or an Amazon S3 prefix.

For more information, see [Configure IAM Roles for EMRFS Requests to Amazon S3 \(p. 164\)](#).

Specifying IAM Roles for EMRFS Using the AWS CLI

The following is an example JSON snippet for specifying custom IAM roles for EMRFS within a security configuration. It demonstrates role mappings for the three different identifier types, followed by a parameter reference.

```
{
    "AuthorizationConfiguration": {
        "EmrFsConfiguration": {
            "RoleMappings": [
                {
                    "Role": "arn:aws:iam::123456789101:role/allow_EMRFS_access_for_user1",
                    "IdentifierType": "User",
                    "Identifiers": [ "user1" ]
                },
                {
                    "Role": "arn:aws:iam::123456789101:role/allow_EMRFS_access_to_MyBuckets",
                    "IdentifierType": "Prefix",
                    "Identifiers": [ "s3://MyBucket/", "s3://MyOtherBucket/" ]
                },
                {
                    "Role": "arn:aws:iam::123456789101:role/allow_EMRFS_access_for_AdminGroup",
                    "IdentifierType": "Group",
                    "Identifiers": [ "AdminGroup" ]
                }
            ]
        }
    }
}
```

Parameter	Description
"AuthorizationConfiguration":	Required.
"EmrFsConfiguration":	Required. Contains role mappings.
"RoleMappings":	Required. Contains one or more role mapping definitions. Role mappings are evaluated in the top-down order that they appear. If a role mapping evaluates as true for an EMRFS call for data in Amazon S3, no further role mappings are evaluated and EMRFS uses the specified IAM role for the request. Role mappings consist of the following required parameters:
"Role":	Specifies the ARN identifier of an IAM role in the format <code>arn:aws:iam::account-id:role/role-name</code> . This is the IAM role that Amazon EMR assumes if the EMRFS request to Amazon S3 matches any of the <code>Identifiers</code> specified.
"IdentifierType":	Can be one of the following: <ul style="list-style-type: none"> "User" specifies that the identifiers are one or more Hadoop users, which can be Linux

Parameter	Description
	<p>account users or Kerberos principals. When the EMRFS request originates with the user or users specified, the IAM role is assumed.</p> <ul style="list-style-type: none"> • "Prefix" specifies that the identifier is an Amazon S3 location. The IAM role is assumed for calls to the location or locations with the specified prefixes. For example, the prefix <code>s3://mybucket/</code> matches <code>s3://mybucket/mydir</code> and <code>s3://mybucket/yetanotherdir</code>. • "Group" specifies that the identifiers are one or more Hadoop groups. The IAM role is assumed if the request originates from a user in the specified group or groups.
"Identifiers":	Specifies one or more identifiers of the appropriate identifier type. Separate multiple identifiers by commas with no spaces.

Specify a Security Configuration for a Cluster

You can specify encryption settings when you create a cluster by specifying the security configuration. You can use the AWS Management Console or the AWS CLI.

Specifying a Security Configuration Using the Console

When using the AWS console to create an Amazon EMR cluster, you choose the security configuration during **Step 4: Security** of the advanced options creation process.

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**, **Go to advanced options**.
3. On the **Step 1: Software and Steps** screen, from the **Release** list, choose **emr-4.8.0** or a more recent release. Choose the settings you want and choose **Next**.
4. On the **Step 2: Hardware** screen, choose the settings you want and choose **Next**. Do the same for **Step 3: General Cluster Settings**.
5. On the **Step 4: Security** screen, under **Encryption Options**, choose a value for **Security configuration**.
6. Configure other security options as desired and choose **Create cluster**.

Specifying a Security Configuration Using the CLI

When you use `aws emr create-cluster`, you can optionally apply a security configuration using `--security-configuration MySecConfig`, where *MySecConfig* is the name of the security configuration, as shown in the following example. The `--release-label` specified must be 4.8.0 or later and the `--instance-type` can be any available.

```
aws emr create-cluster --instance-type m4.large --release-label emr-5.0.0 --security-configuration mySecConfig
```

Configure IAM Roles for Amazon EMR Permissions to AWS Services

Amazon EMR and applications such as Hadoop and Spark need permissions to access other AWS resources and perform actions when they run. Each cluster in Amazon EMR must have a *service role* and a role for the Amazon EC2 *instance profile*. The IAM policies attached to these roles provide permissions for the cluster to interoperate with other AWS services on behalf of a user.

All clusters in all regions require a service role and a role for the EC2 instance profile. An additional role, the Auto Scaling role, is required if your cluster uses automatic scaling. For more information, see [IAM Roles](#) and [Using Instance Profiles](#) in the *IAM User Guide*.

If you are creating a cluster for the first time in an account, roles for Amazon EMR do not yet exist. You can specify default roles for Amazon EMR to create, or you can create custom roles and specify them when you create a cluster. For more information, see [Use Default IAM Roles and Managed Policies \(p. 188\)](#), and [Customize IAM Roles \(p. 194\)](#).

Amazon EMR uses the following roles when interacting with other AWS services.

EMR Role

The EMR role defines the allowable actions for Amazon EMR when provisioning resources and performing other service-level tasks that are not performed in the context of an EC2 instance running within a cluster. The default role is `EMR_DefaultRole`.

EMR Role for EC2

The EMR role for EC2 is used by EC2 instances within the cluster. In other words, this is the role associated with the EC2 instance profile for cluster instances. The permissions associated with this role apply to processes that run on cluster instances. As long as an application process runs on top of the Hadoop ecosystem, the application assumes this role to interact with other AWS services. The default role is `EMR_EC2_DefaultRole`.

Note

If you create a custom EMR role for EC2, follow the basic work flow, which automatically creates an instance profile of the same name. Amazon EC2 allows you to create instance profiles and roles with different names, but Amazon EMR does not support this configuration, and it results in an "invalid instance profile" error when you create the cluster.

Clusters typically read and write data to Amazon S3 using EMRFS. EMRFS uses this role by default. When you have multiple cluster users and multiple data stores, you may want users to have different permissions to EMRFS data in Amazon S3. To do this, you can set up a security configuration with IAM roles for EMRFS requests to Amazon S3. EMRFS can assume different roles with different permissions policies, based on the user or group making the request or the location of EMRFS data in Amazon S3. For requests that match the criteria that you specify, EMRFS uses the corresponding roles instead of the EMR role for EC2. If no match is found, EMRFS falls back to the EMR role for EC2. For more information, see [Configure IAM Roles for EMRFS Requests to Amazon S3 \(p. 164\)](#).

If you have application code on your cluster that calls AWS services directly, you may need to use the SDK to specify roles. For more information, see [Use IAM Roles with Applications That Call AWS Services Directly \(p. 196\)](#).

Automatic Scaling Role

The automatic scaling role for EMR performs a similar function as the service role, but allows additional actions for dynamically scaling environments. The default role is `EMR_AutoScaling_DefaultRole`.

Service-Linked Role

EMR automatically creates a service-linked role. If the service for Amazon EMR has lost the ability to clean up Amazon EC2 resources, Amazon EMR can use this role to do so. For more information, see [Using the Service-Linked Role for Amazon EMR \(p. 197\)](#). The EMR role must be allowed to create a service-linked role for Spot Instances.

The `AmazonElasticMapReduceFullAccess`, which is the default managed policy for users to have full permissions for Amazon EMR, includes a statement that allows the `iam:PassRole` permissions for all resources. This statement allows the user to pass any role to other AWS services so that Amazon EMR can interact with those services on behalf of the user.

To implement a more restrictive policy, attach an inline policy to appropriate users or groups that allows `iam:PassRole` only for roles specific to Amazon EMR. The following example demonstrates a statement that allows `iam:PassRole` permissions only for the default Amazon EMR roles: `EMR_DefaultRole`, `EMR_EC2_DefaultRole`, and `EMR_AutoScalingDefaultRole`. If you use custom roles, replace the default role names with your custom role names.

```
{  
    "Action": "iam:PassRole",  
    "Effect": "Allow",  
    "Resource": [  
        "arn:aws:iam::*:role/EMR_DefaultRole",  
        "arn:aws:iam::*:role/EMR_EC2_DefaultRole",  
        "arn:aws:iam::*:role/EMR_AutoScaling_DefaultRole"  
    ]  
}
```

Topics

- [Use Default IAM Roles and Managed Policies \(p. 188\)](#)
- [Allow Users and Groups to Create and Modify Roles \(p. 194\)](#)
- [Customize IAM Roles \(p. 194\)](#)
- [Use Resource-Based Policies for Amazon EMR Access to AWS Glue Data Catalog \(p. 196\)](#)
- [Use IAM Roles with Applications That Call AWS Services Directly \(p. 196\)](#)
- [Using the Service-Linked Role for Amazon EMR \(p. 197\)](#)

Use Default IAM Roles and Managed Policies

Amazon EMR provides the default roles listed below. The managed policies listed are created and attached to them by default. These roles and policies do not exist in your account until you create them. After you create them, you can view the roles, the policies attached to them, and the permissions allowed or denied by the policies in the IAM console (<https://console.aws.amazon.com/iam/>). Managed policies are created and maintained by AWS, so they are updated automatically if service requirements change.

Default Role	Description	Default Managed Policy
<code>EMR_DefaultRole</code>	This is the EMR role, which allows Amazon EMR to call other AWS services such as Amazon EC2 on your behalf.	<code>AmazonElasticMapReduceRole</code> Important Requesting Spot Instances requires a service-linked role. If this role doesn't exist, the Amazon EMR

Default Role	Description	Default Managed Policy
		<p>service role must have permissions to create it or a permission error occurs. The managed policy includes a statement to allow this action. If you customize this role or policy, be sure to include a statement that allows the creation of this service-linked role. For more information, see Default Contents of AmazonElasticMapReduceRole Permissions Policy (p. 191) and Service-Linked Role for Spot Instance Requests in the Amazon EC2 User Guide for Linux Instances.</p>
<code>EMR_EC2_DefaultRole</code>	<p>The EMR role for EC2 instances within a cluster. Processes that run on cluster instances use this role when they call other AWS services. For accessing EMRFS data in Amazon S3, you can specify different roles to be assumed based on the user or group making the request, or on the location of data in Amazon S3. For more information, see Configure IAM Roles for EMRFS Requests to Amazon S3 (p. 164).</p>	<p>AmazonElasticMapReduceforEC2Role. For more information, see Default Contents of AmazonElasticMapReduceforEC2Role Permissions Policy (p. 192).</p>
<code>EMR_AutoScaling_DefaultRole</code>	<p>Allows additional actions for dynamically scaling environments. Required only for clusters that use automatic scaling in Amazon EMR. For more information, see Using Automatic Scaling in Amazon EMR (p. 254).</p>	<p>AmazonElasticMapReduceforAutoScalingRole. For more information, see Default Contents of AmazonMapReduceforAutoScalingRole Permissions Policy (p. 193).</p>

To create default IAM roles for Amazon EMR in your account for the first time

1. If you are logged in as an IAM user, make sure that you are allowed the appropriate IAM actions to create roles. For more information, see [Allow Users and Groups to Create and Modify Roles \(p. 194\)](#).

2. Use the Amazon EMR console to create a cluster and leave the default roles specified. You can do this either using **Quick options** or using **Advanced options**. Amazon EMR automatically creates the roles when it launches the cluster. They are available to any clusters that you launch later.

—OR—

Use the `emr create-default-roles` command from the AWS CLI.

The output of the command lists the contents of the roles. The CLI configuration file (`config`) is populated with these role names for the `service_role` and `instance_profile` values. For example:

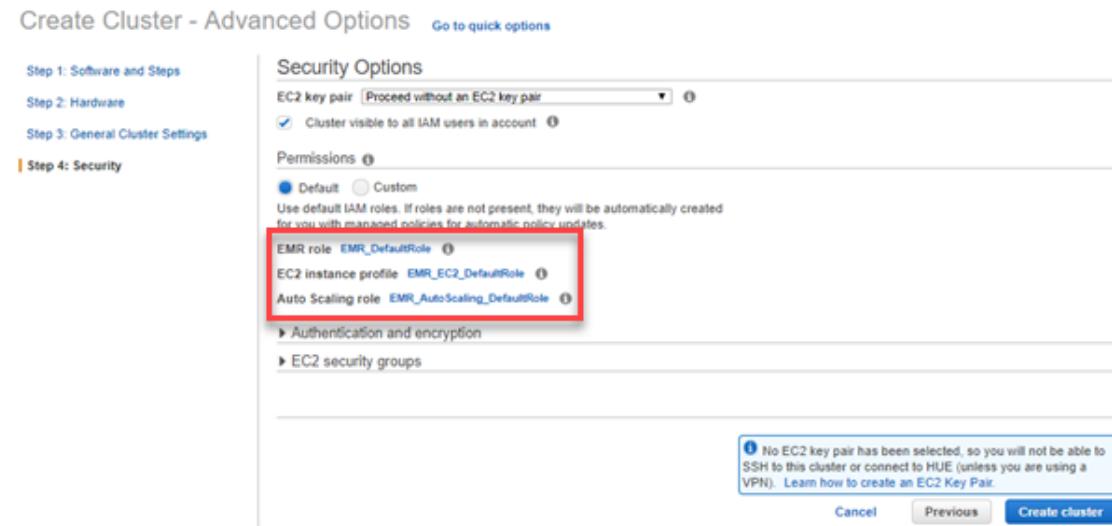
```
[default]
output = json
region = us-west-1
aws_access_key_id = myAccessKeyID
aws_secret_access_key = mySecretAccessKey
emr =
    service_role = EMR_DefaultRole
    instance_profile = EMR_EC2_DefaultRole
```

Managed Policy Contents

The contents of managed policies for Amazon EMR default roles are shown below. Managed policies are automatically updated, so the policies shown here may be out-of-date. Use the procedure below to view the most recent policy. You can also check the version listed below against the version listed for each managed policy.

To view current default roles and managed policies for Amazon EMR

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**, **Go to advanced options**.
3. Choose **Next** until you reach **Security options**.
4. Choose the service role you want to view.



5. Choose the permissions policy to view the policy JSON.

The screenshot shows the AWS IAM Roles page. On the left, there's a sidebar with links like Dashboard, Groups, Users, Roles (which is selected), Policies, Identity providers, Account settings, Credential report, and Encryption keys. The main area is titled 'Roles > EMR_DefaultRole' and has a 'Summary' tab selected. It displays the Role ARN (arn:aws:iam::210774411744:role/EMR_DefaultRole), Role description (empty), Instance Profile ARNs (empty), Path (/), and Creation time (2015-04-24 13:42 PST). Below this is a navigation bar with tabs: Permissions (selected), Trust relationships, Access Advisor, and Revoke sessions. Under the 'Permissions' tab, there's a button 'Attach policy' and a table showing 'Attached policies: 1'. The table has columns for Policy name and Policy type. A single row is shown: 'AmazonElasticMapReduceRole' (AWS managed policy). This row is highlighted with a red box. At the bottom right of the table is a link 'Add inline policy'.

6. Choose **Policy versions** to see the current policy version number, a history of policy updates, and the policy JSON for past versions.

Default Contents of AmazonElasticMapReduceRole Permissions Policy

The contents of version 9 of this policy are shown below.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Resource": "*",
      "Action": [
        "ec2:AuthorizeSecurityGroupEgress",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:CancelSpotInstanceRequests",
        "ec2>CreateNetworkInterface",
        "ec2>CreateSecurityGroup",
        "ec2>CreateTags",
        "ec2>DeleteNetworkInterface",
        "ec2>DeleteSecurityGroup",
        "ec2>DeleteTags",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeImages",
        "ec2:DescribeInstanceState",
        "ec2:DescribeInstances",
        "ec2:DescribeKeyPairs",
        "ec2:DescribeNetworkAcls",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribePrefixLists",
        "ec2:DescribeRouteTables",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSpotInstanceRequests",
        "ec2:DescribeSpotPriceHistory",
        "ec2:DescribeSubnets",
        "ec2:DescribeTags",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeVpcEndpoints",
        "ec2:DescribeVpcEndpointServices",
        "ec2:DescribeVpcEndpointStatus"
      ]
    }
  ]
}
```

```

        "ec2:DescribeVpcs",
        "ec2:DetachNetworkInterface",
        "ec2:ModifyImageAttribute",
        "ec2:ModifyInstanceState",
        "ec2:RequestSpotInstances",
        "ec2:RevokeSecurityGroupEgress",
        "ec2:RunInstances",
        "ec2:TerminateInstances",
        "ec2:DeleteVolume",
        "ec2:DescribeVolumeStatus",
        "ec2:DescribeVolumes",
        "ec2:DetachVolume",
        "iam:GetRole",
        "iam:GetRolePolicy",
        "iam>ListInstanceProfiles",
        "iam>ListRolePolicies",
        "iam:PassRole",
        "s3>CreateBucket",
        "s3:Get*",
        "s3>List*",
        "sdb:BatchPutAttributes",
        "sdb:Select",
        "sns>CreateQueue",
        "sns>Delete*",
        "sns:GetQueue*",
        "sns:PurgeQueue",
        "sns:ReceiveMessage",
        "cloudwatch:PutMetricAlarm",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:DeleteAlarms",
        "application-autoscaling:RegisterScalableTarget",
        "application-autoscaling:DeregisterScalableTarget",
        "application-autoscaling:PutScalingPolicy",
        "application-autoscaling:DeleteScalingPolicy",
        "application-autoscaling:Describe"
    ],
},
{
    "Effect": "Allow",
    "Action": "iam>CreateServiceLinkedRole",
    "Resource": "arn:aws:iam::*:role/aws-service-role/spot.amazonaws.com/AWSServiceRoleForEC2Spot*",
    "Condition": {
        "StringLike": {
            "iam:AWSServiceName": "spot.amazonaws.com"
        }
    }
}
]
}

```

Default Contents of AmazonElasticMapReduceforEC2Role Permissions Policy

The contents of version 3 of this policy are shown below.

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

```

```
    "ec2:Describe*",
    "elasticmapreduce:Describe*",
    "elasticmapreduce>ListBootstrapActions",
    "elasticmapreduce>ListClusters",
    "elasticmapreduce>ListInstanceGroups",
    "elasticmapreduce>ListInstances",
    "elasticmapreduce>ListSteps",
    "kinesis>CreateStream",
    "kinesis>DeleteStream",
    "kinesis>DescribeStream",
    "kinesis>GetRecords",
    "kinesis>GetShardIterator",
    "kinesis>MergeShards",
    "kinesis>PutRecord",
    "kinesis>SplitShard",
    "rds:Describe*",
    "s3:*",
    "sdb:*",
    "sns:*",
    "sqs:*",
    "glue>CreateDatabase",
    "glue>UpdateDatabase",
    "glue>DeleteDatabase",
    "glue>GetDatabase",
    "glue>GetDatabases",
    "glue>CreateTable",
    "glue>UpdateTable",
    "glue>DeleteTable",
    "glue>GetTable",
    "glue>GetTables",
    "glue>GetTableVersions",
    "glue>CreatePartition",
    "glue>BatchCreatePartition",
    "glue>UpdatePartition",
    "glue>DeletePartition",
    "glue>BatchDeletePartition",
    "glue>GetPartition",
    "glue>GetPartitions",
    "glue>BatchGetPartition",
    "glue>CreateUserDefinedFunction",
    "glue>UpdateUserDefinedFunction",
    "glue>DeleteUserDefinedFunction",
    "glue GetUserDefinedFunction",
    "glue GetUserDefinedFunctions"
]
}
]
}
```

Default Contents of AmazonMapReduceforAutoScalingRole Permissions Policy

The contents of version 1 of this policy are shown below.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "cloudwatch:DescribeAlarms",
        "elasticmapreduce>ListInstanceGroups",
        "elasticmapreduce>ModifyInstanceGroups"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

```
    ]  
}
```

Allow Users and Groups to Create and Modify Roles

IAM principals (users and groups) who create, modify, and specify roles for a cluster, including default roles, must be allowed to perform the following actions. For details about each action, see [Actions](#) in the [IAM API Reference](#).

- `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 the creation of the default roles.

Customize IAM Roles

You may want to customize IAM roles and permissions for your requirements. For example, if your application does not use EMRFS consistent view, you may not want to allow Amazon EMR to access Amazon DynamoDB. To customize permissions, we recommend that you create new roles and policies. Begin with the permissions in the managed policies for the default roles (for example, `AmazonElasticMapReduceforEC2Role` and `AmazonElasticMapReduceRole`). Then, copy and paste the contents to new policy statements, modify the permissions as appropriate, and attach the modified permissions policies to the roles that you create. You must have the appropriate IAM permissions to work with roles and policies. For more information, see [Allow Users and Groups to Create and Modify Roles \(p. 194\)](#).

If you create a custom EMR role for EC2, follow the basic work flow, which automatically creates an instance profile of the same name. Amazon EC2 allows you to create instance profiles and roles with different names, but Amazon EMR does not support this configuration, and it results in an "invalid instance profile" error when you create the cluster.

Important

Inline policies are not automatically updated when service requirements change. If you create and attach inline policies, be aware that service updates might occur that suddenly cause permissions errors. For more information, see [Managed Policies and Inline Policies](#) in the [IAM User Guide](#) and [Specify Custom IAM Roles When You Create a Cluster \(p. 195\)](#).

For more information about working with IAM roles, see the following topics in the [IAM User Guide](#):

- [Creating a Role to Delegate Permissions to an AWS Service](#)
- [Modifying a Role](#)
- [Deleting a Role](#)

Specify Custom IAM Roles When You Create a Cluster

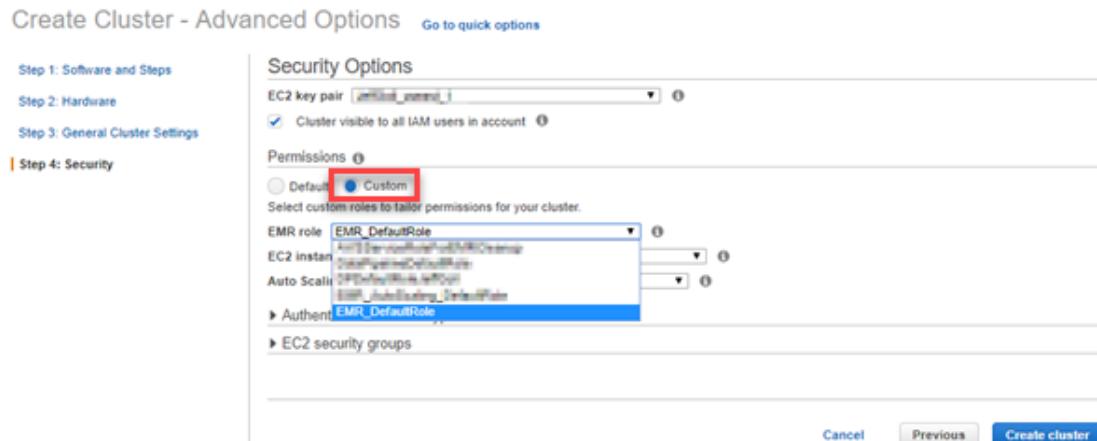
You specify the service role for Amazon EMR and the role for the Amazon EC2 instance profile when you create a cluster. The user who is creating clusters needs permissions to retrieve and assign roles to Amazon EMR and EC2 instances. Otherwise, a **User account is not authorized to call EC2** error occurs. For more information, see [Allow Users and Groups to Create and Modify Roles \(p. 194\)](#).

Use the Console to Specify Custom Roles

When you create a cluster, you can specify a custom service role for Amazon EMR, a custom role for the EC2 instance profile, and a custom Auto Scaling role using **Advanced options**. When you use **Quick options**, the default service role and the default role for the EC2 instance profile are specified. For more information, see [Use Default IAM Roles and Managed Policies \(p. 188\)](#).

To specify custom IAM roles 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. Choose the cluster settings appropriate for your application until you reach **Security Options**.
Under **Permissions**, the **Default** roles for Amazon EMR are selected.
4. Choose **Custom**.
5. For each role type, select a role from the list. Only roles within your account that have the appropriate trust policy for that role type are listed.



6. Choose other options as appropriate for your cluster and then choose **Create Cluster**.

Use the AWS CLI to Specify Custom Roles

You can specify a service role and a role for the EC2 instance profile using options with the `create-cluster` command from the AWS CLI. Use the `--service-role` option to specify the service role. Use the `InstanceProfile` argument of the `--ec2-attributes` option to specify the role for the EC2 instance profile.

You can use these options to specify default roles explicitly rather than using the `--use-default-roles` option. The `--use-default-roles` option specifies the service role and the role for the EC2 instance profile defined in the CLI configuration file, which you can update to reflect your custom roles. For more information, see [Use Default IAM Roles and Managed Policies \(p. 188\)](#).

The Auto Scaling role is specified using a separate option, `--auto-scaling-role`. For more information, see [Using Automatic Scaling in Amazon EMR \(p. 254\)](#).

To specify custom IAM roles using the AWS CLI

- The following command specifies the custom service role, `MyEMRServiceRole`, and a custom role for the EC2 instance profile, `MyEC2RoleForEMR`, when launching a cluster. This example uses the default Amazon EMR role.

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-5.20.0 \
--applications Name=Hive Name=Pig --service-role MyEMRServiceRole \
--ec2-attributes InstanceProfile=MyEC2RoleForEMR, \
KeyName=myKey --instance-type m4.large --instance-count 3
```

Use Resource-Based Policies for Amazon EMR Access to AWS Glue Data Catalog

If you use AWS Glue in conjunction with Hive, Spark, or Presto in Amazon EMR, AWS Glue supports resource-based policies to control access to Data Catalog resources. These resources include databases, tables, connections, and user-defined functions. For more information, see [AWS Glue Resource Policies](#) in the [AWS Glue Developer Guide](#).

When using resource-based policies to limit access to AWS Glue from within Amazon EMR, the principal that you specify in the permissions policy must be the role ARN associated with the EC2 instance profile that is specified when a cluster is created. For example, if you use the default EC2 instance profile, this role ARN has the following format:

```
arn:aws:iam::acct-id:role/EMR_EC2_DefaultRole
```

The `acct-id` can be different from the AWS Glue account ID. This enables access from EMR clusters in different accounts. You can specify multiple principals, each from a different account.

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.

Using the Service-Linked Role for Amazon EMR

Amazon EMR uses AWS Identity and Access Management (IAM) [service-linked roles](#). A service-linked role is a unique type of IAM role that is linked directly to Amazon EMR. The service-linked role is predefined by Amazon EMR and includes the permissions that Amazon EMR requires to call Amazon EC2 on your behalf to clean up cluster resources after they are no longer in use. The service-linked role works together with the Amazon EMR service role and Amazon EC2 instance profile for Amazon EMR. For more information about the service role and instance profile, see [Configure IAM Roles for Amazon EMR Permissions to AWS Services \(p. 187\)](#).

Amazon EMR defines the permissions of this service-linked role, and unless defined otherwise, only Amazon EMR can assume the role. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity. You can delete the role only after you terminate all EMR clusters in the account.

For information about other services that support service-linked roles, see [AWS Services That Work with IAM](#) and look for the services that have **Yes** in the **Service-Linked Role** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Service-Linked Role Permissions for Amazon EMR

Amazon EMR uses the **AWSServiceRoleForEMRCleanup** role, which is a service-based role that allows Amazon EMR to terminate and delete Amazon EC2 resources on your behalf if the Amazon EMR service role has lost that ability. Amazon EMR creates the role automatically during cluster creation if it does not already exist.

The AWSServiceRoleForEMRCleanup service-linked role trusts the following services to assume the role:

- elasticmapreduce.amazonaws.com

The AWSServiceRoleForEMRCleanup service-linked role permissions policy allows Amazon EMR to complete the following actions on the specified resources:

- Action: `DescribeInstances` on ec2
- Action: `DescribeSpotInstanceRequests` on ec2
- Action: `ModifyInstanceState` on ec2
- Action: `TerminateInstances` on ec2
- Action: `CancelSpotInstanceRequests` on ec2
- Action: `DeleteNetworkInterface` on ec2
- Action: `DescribeInstanceAttribute` on ec2
- Action: `DescribeVolumeStatus` on ec2
- Action: `DescribeVolumes` on ec2
- Action: `DetachVolume` on ec2
- Action: `DeleteVolume` on ec2

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role.

To allow an IAM entity to create the AWSServiceRoleForEMRCleanup service-linked role

Add the following statement to the permissions policy for the IAM entity that needs to create the service-linked role:

```
{  
    "Effect": "Allow",  
    "Action": [  
        "iam:CreateServiceLinkedRole",  
        "iam:PutRolePolicy"  
    ],  
    "Resource": "arn:aws:iam::*:role/aws-service-role/elasticmapreduce.amazonaws.com*/  
AWSServiceRoleForEMRCleanup*",  
    "Condition": {  
        "StringLike": {  
            "iam:AWSServiceName": [  
                "elasticmapreduce.amazonaws.com",  
                "elasticmapreduce.amazonaws.com.cn"  
            ]  
        }  
    }  
}
```

To allow an IAM entity to edit the description of the AWSServiceRoleForEMRCleanup service-linked role

Add the following statement to the permissions policy for the IAM entity that needs to edit the description of a service-linked role:

```
{  
    "Effect": "Allow",  
    "Action": [  
        "iam:UpdateRoleDescription"  
    ],  
    "Resource": "arn:aws:iam::*:role/aws-service-role/elasticmapreduce.amazonaws.com*/  
AWSServiceRoleForEMRCleanup*",
```

```
"Condition": {  
    "StringLike": {  
        "iam:AWSServiceName": [  
            "elasticmapreduce.amazonaws.com",  
            "elasticmapreduce.amazonaws.com.cn"  
        ]  
    }  
}
```

To allow an IAM entity to delete the AWSServiceRoleForEMRCleanup service-linked role

Add the following statement to the permissions policy for the IAM entity that needs to delete a service-linked role:

```
{  
    "Effect": "Allow",  
    "Action": [  
        "iam:DeleteServiceLinkedRole",  
        "iam:GetServiceLinkedRoleDeletionStatus"  
    ],  
    "Resource": "arn:aws:iam::*:role/aws-service-role/elasticmapreduce.amazonaws.com*/  
AWSServiceRoleForEMRCleanup*",  
    "Condition": {  
        "StringLike": {  
            "iam:AWSServiceName": [  
                "elasticmapreduce.amazonaws.com",  
                "elasticmapreduce.amazonaws.com.cn"  
            ]  
        }  
    }  
}
```

Creating a Service-Linked Role for Amazon EMR

You don't need to manually create the AWSServiceRoleForEMRCleanup role. When you launch a cluster, either for the first time or when a service-linked role is not present, Amazon EMR creates the service-linked role for you. You must have permissions to create the service-linked role. For an example statement that adds this capability to the permissions policy of an IAM entity (such as a user, group, or role), see [Service-Linked Role Permissions for Amazon EMR \(p. 197\)](#).

Important

If you were using Amazon EMR before October 24, 2017, when service-linked roles were not supported, then Amazon EMR created the AWSServiceRoleForEMRCleanup role in your account. For more information, see [A New Role Appeared in My IAM Account](#).

Editing a Service-Linked Role for Amazon EMR

Amazon EMR does not allow you to edit the AWSServiceRoleForEMRCleanup service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM.

Editing a Service-Linked Role Description (IAM Console)

You can use the IAM console to edit the description of a service-linked role.

To edit the description of a service-linked role (console)

1. In the navigation pane of the IAM console, choose **Roles**.

2. Choose the name of the role to modify.
3. To the far right of **Role description**, choose **Edit**.
4. Type a new description in the box and choose **Save**.

Editing a Service-Linked Role Description (IAM CLI)

You can use IAM commands from the AWS Command Line Interface to edit the description of a service-linked role.

To change the description of a service-linked role (CLI)

1. (Optional) To view the current description for a role, use the following commands:

```
$ aws iam get-role --role-name role-name
```

Use the role name, not the ARN, to refer to roles with the CLI commands. For example, if a role has the following ARN: `arn:aws:iam::123456789012:role/myrole`, you refer to the role as **myrole**.

2. To update a service-linked role's description, use one of the following commands:

```
$ aws iam update-role-description --role-name role-name --description description
```

Editing a Service-Linked Role Description (IAM API)

You can use the IAM API to edit the description of a service-linked role.

To change the description of a service-linked role (API)

1. (Optional) To view the current description for a role, use the following command:

IAM API: [GetRole](#)

2. To update a role's description, use the following command:

IAM API: [UpdateRoleDescription](#)

Deleting a Service-Linked Role for Amazon EMR

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way, you don't have an unused entity that is not being actively monitored or maintained. However, you must clean up your service-linked role before you can delete it.

Cleaning Up a Service-Linked Role

Before you can use IAM to delete a service-linked role, you must first confirm that the role has no active sessions and remove any resources used by the role.

To check whether the service-linked role has an active session in the IAM console

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Roles**. Select the name (not the check box) of the `AWSServiceRoleForEMRCleanup` role.

3. On the **Summary** page for the selected role, choose **Access Advisor**.
4. On the **Access Advisor** tab, review the recent activity for the service-linked role.

Note

If you are unsure whether Amazon EMR is using the AWSServiceRoleForEMRCleanup role, you can try to delete the role. If the service is using the role, then the deletion fails and you can view the regions where the role is being used. If the role is being used, then you must wait for the session to end before you can delete the role. You cannot revoke the session for a service-linked role.

To remove Amazon EMR resources used by the AWSServiceRoleForEMRCleanup

- Terminate all clusters in your account. For more information, see [Terminate a Cluster \(p. 251\)](#).

Deleting a Service-Linked Role (IAM Console)

You can use the IAM console to delete a service-linked role.

To delete a service-linked role (console)

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Roles**. Select the check box next to AWSServiceRoleForEMRCleanup, not the name or row itself.
3. For **Role actions** at the top of the page, choose **Delete role**.
4. In the confirmation dialog box, review the service last accessed data, which shows when each of the selected roles last accessed an AWS service. This helps you to confirm whether the role is currently active. To proceed, choose **Yes, Delete**.
5. Watch the IAM console notifications to monitor the progress of the service-linked role deletion. Because the IAM service-linked role deletion is asynchronous, after you submit the role for deletion, the deletion task can succeed or fail. If the task fails, you can choose **View details** or **View Resources** from the notifications to learn why the deletion failed. If the deletion fails because there are resources in the service that are being used by the role, then the reason for the failure includes a list of resources.

Deleting a Service-Linked Role (IAM CLI)

You can use IAM commands from the AWS Command Line Interface to delete a service-linked role. Because a service-linked role cannot be deleted if it is being used or has associated resources, you must submit a deletion request. If these conditions are not met, that request can be denied.

To delete a service-linked role (CLI)

1. To check the status of the deletion task, you must capture the `deletion-task-id` from the response. Type the following command to submit a service-linked role deletion request:

```
$ aws iam delete-service-linked-role --role-name AWSServiceRoleForEMRCleanup
```

2. Type the following command to check the status of the deletion task:

```
$ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
```

The status of the deletion task can be `NOT_STARTED`, `IN_PROGRESS`, `SUCCEEDED`, or `FAILED`. If the deletion fails, the call returns the reason that it failed so that you can troubleshoot.

Deleting a Service-Linked Role (IAM API)

You can use the IAM API to delete a service-linked role. Because a service-linked role cannot be deleted if it is being used or has associated resources, you must submit a deletion request. If these conditions are not met, that request can be denied.

To delete a service-linked role (API)

1. To submit a deletion request for a service-linked role, call [DeleteServiceLinkedRole](#). In the request, specify the `AWSServiceRoleForEMRCleanup` role name.
To check the status of the deletion task, you must capture the `DeletionTaskId` from the response.
2. To check the status of the deletion, call [GetServiceLinkedRoleDeletionStatus](#). In the request, specify the `DeletionTaskId`.

The status of the deletion task can be `NOT_STARTED`, `IN_PROGRESS`, `SUCCEEDED`, or `FAILED`. If the deletion fails, the call returns the reason that it failed so that you can troubleshoot.

Supported Regions for Amazon EMR Service-Linked Roles

Amazon EMR supports using service-linked roles in the following regions.

Region name	Region identity	Support in Amazon EMR
US East (N. Virginia)	us-east-1	Yes
US East (Ohio)	us-east-2	Yes
US West (N. California)	us-west-1	Yes
US West (Oregon)	us-west-2	Yes
Asia Pacific (Mumbai)	ap-south-1	Yes
Asia Pacific (Osaka-Local)	ap-northeast-3	Yes
Asia Pacific (Seoul)	ap-northeast-2	Yes
Asia Pacific (Singapore)	ap-southeast-1	Yes
Asia Pacific (Sydney)	ap-southeast-2	Yes
Asia Pacific (Tokyo)	ap-northeast-1	Yes
Canada (Central)	ca-central-1	Yes
EU (Frankfurt)	eu-central-1	Yes
EU (Ireland)	eu-west-1	Yes
EU (London)	eu-west-2	Yes
EU (Paris)	eu-west-3	Yes
South America (São Paulo)	sa-east-1	Yes

Manage Clusters

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 \(p. 203\)](#)
- [Connect to the Cluster \(p. 238\)](#)
- [Terminate a Cluster \(p. 251\)](#)
- [Scaling Cluster Resources \(p. 253\)](#)
- [Cloning a Cluster Using the Console \(p. 269\)](#)
- [Submit Work to a Cluster \(p. 270\)](#)
- [Automate Recurring Clusters with AWS Data Pipeline \(p. 274\)](#)

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 Status and Details \(p. 203\)](#)
- [Enhanced Step Debugging \(p. 208\)](#)
- [View Application History \(p. 210\)](#)
- [View Log Files \(p. 211\)](#)
- [View Cluster Instances in Amazon EC2 \(p. 215\)](#)
- [CloudWatch Events and Metrics \(p. 216\)](#)
- [View Cluster Application Metrics with Ganglia \(p. 236\)](#)
- [Logging Amazon EMR API Calls in AWS CloudTrail \(p. 236\)](#)

View Cluster Status and Details

After you create a cluster, you can monitor its status and get detailed information about its execution and errors that may have occurred, even after it has terminated. Amazon EMR saves metadata about terminated clusters for your reference for two months, after which the metadata is deleted. Application history is saved for one week from the time it is recorded, regardless of whether the cluster is running or terminated. You can't delete clusters from the cluster history, but using the AWS Management Console, you can use the **Filter**, and using the AWS CLI, you can use options with the `list-clusters` command to focus on the clusters that you care about.

View Cluster Status Using the AWS Management Console

The **Clusters List** in the [Amazon EMR console](#) lists all the clusters in your account and AWS Region, including terminated clusters. The list shows the following for each cluster: the **Name** and **ID**, the **Status**, the **Creation time**, the **Elapsed time** that the cluster was running, and the **Normalized instance hours**.

that have accrued for all EC2 instances in the cluster. This list is the starting point for monitoring the status of your clusters. It's designed so that you can drill down into each cluster's details for analysis and troubleshooting.

To view an abridged summary of cluster information

- Select the down arrow next to the link for the cluster under **Name**.

The cluster's row expands to provide more information about the cluster, hardware, steps, and bootstrap actions. Use the links in this section to drill into specifics. For example, click a link under **Steps** to access step log files, see the JAR associated with the step, drill into the step's jobs and tasks, and access log files.

Filter:		All clusters	Filter loaded clusters ...	100 clusters loaded	load more	
	Name	ID	Status	Creation time (UTC-7) ▾	Elapsed time	Normal instances
	Word count	j-3HKR4JT224DZZ	Terminated All steps completed	2014-04-21 16:01	10 minutes	3
Summary			Steps		View all interactive jobs	
Master public DNS:	ec2-54-84-190-14.compute-1.amazonaws.com		Name	Status	Start time (UTC-7) ▾	Elapsed time
Termination protection:	On		Word count	Completed	2014-04-21 16:06	2 minutes
Tags:	--		Setup hadoop debugging	Completed	2014-04-21 16:05	37 seconds
Hardware						No bootstrap actions available
Master: Terminated 1 m1.small						
Core: Terminated 2 m1.small						
Task: --						

To view cluster status in depth

- Choose the cluster link under **Name** to open a cluster details page for the cluster. Use each tab to view information as described in the following section.

Use each tab for the following information:

Summary	Application history	Monitoring	Events	Steps	Configurations	Bootstrap actions
Connections: Enable Web Connection – Spark History Server, Resource Manager ... (View All) Master public DNS: http://54.84.190.14.compute-1.amazonaws.com SSH Tags: View All / Edit	Summary ID: j-162692D2BOSWL Creation date: 2017-09-07 13:42 (UTC-7) Elapsed time: 10 days Auto-terminate: No Termination On Change protection: View Custom AMI ID: --	Configuration details Release label: emr-5.8.0 Hadoop distribution: Amazon Applications: Spark 2.0 Log URI: https://aws.amazon.com/reviews/j-162692D2BOSWL/162692D2BOSWL EMRFS consistent view: View	Network and hardware Availability zone: us-east-1b Subnet ID: -- Master: Running 1 m3.xlarge Core: Running 2 m3.xlarge Task: Running 1 m3.xlarge	Security and access Key name: MyKeyPair EC2 instance profile: EMR_EC2_DefaultRole EMR role: EMR_DefaultRole Visible to all users: All Change Security groups for sg-01f97fba (ElasticMapReduce - Master) Security groups for sg-01f97fba (ElasticMapReduce - Core & Task: Slave)		

Tab	Information
Summary	Use this tab to view basics of your cluster configuration, such as the URL to use for SSH connections to the master node, what open-source applications Amazon EMR installed when the cluster was created, where logs are stored in Amazon S3, and what version of Amazon EMR was used to create the cluster.
Application history	Use this tab to view YARN application details. For Spark jobs, you can drill down into available information about jobs, stages, and executors. For more information, see View Application History (p. 210) .
Monitoring	Use this tab to view graphs depicting key indicators of cluster operation over a time

Tab	Information
	period that you specify. You can view cluster-level data, node-level data, and information about I/O and data storage.
Hardware	Use this tab to view information about nodes in your cluster, including EC2 instance IDs, DNS names, and IP addresses, and more.
Events	Use this tab to view the event log for your cluster. For more information, see Monitor CloudWatch Events (p. 216) .
Steps	Use this tab to see the status and access log files for steps that you submitted. For more information about steps, see Work with Steps Using the CLI and Console (p. 270) .
Configurations	Use this tab to view any customized configuration objects applied to the cluster. For more information about configuration classifications, see Configuring Applications in the <i>Amazon EMR Release Guide</i> .
Bootstrap actions	Use this tab to view the status of any bootstrap actions the cluster runs when it launches. Bootstrap actions are used for custom software installations and advanced configuration. For more information, see Create Bootstrap Actions to Install Additional Software (p. 92) .

View Cluster Status Using the AWS CLI

The following examples demonstrate how to retrieve cluster details using the AWS CLI. For more information about available commands, see the [AWS CLI Command Reference for Amazon EMR](#). You can 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. The following example demonstrates using the `describe-cluster` command, followed by examples of the `list-clusters` command.

Example Viewing Cluster Status

To use the `describe-cluster` command, you need the cluster ID. This example demonstrates using to get a list of clusters created within a certain date range, and then using one of the cluster IDs returned to list more information about an individual cluster's status.

The following command describes cluster `j-1K48XXXXXXHCB`, which you replace with your cluster ID.

```
aws emr describe-cluster --cluster-id j-1K48XXXXXXHCB
```

The output of your command 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-cXXXXXX0",
    "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-2EEXAMPLEXXP",
    "Configurations": [],
    "InstanceType": "m4.large",
    "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": "m4.large",
    "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"
    }
]
}
```

Example Listing Clusters by Creation Date

To retrieve clusters created within a specific date range, use the `list-clusters` command with the `--created-after` and `--created-before` parameters.

The following command lists all clusters created between October 09, 2014 and October 12, 2014.

```
aws emr list-clusters --created-after 2014-10-09T00:12:00 --created-
before 2014-10-12T00:12:00
```

Example Listing Clusters by State

To list clusters by state, use the `list-clusters` command with the `--cluster-states` parameter. Valid cluster states include: STARTING, BOOTSTRAPPING, RUNNING, WAITING, TERMINATING, TERMINATED, and TERMINATED_WITH_ERRORS.

```
aws emr list-clusters --cluster-states TERMINATED
```

You can also use the following shortcut parameters to list all clusters in the states specified.:

- `--active` filters clusters in the STARTING, BOOTSTRAPPING, RUNNING, WAITING, or TERMINATING states.
- `--terminated` filters clusters in the TERMINATED state.
- `--failed` parameter filters clusters in the TERMINATED_WITH_ERRORS state.

The following commands return the same result.

```
aws emr list-clusters --cluster-states TERMINATED
```

```
aws emr list-clusters --terminated
```

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.

	ID	Name	Status
●	● s-IVXTXADABLBE	WordCount	Failed
Status: FAILED Reason: Output directory already exists. Log File: s3://aws-logs-us-east-1/elasticmapreduce/j-MQG1MTOTJ7HM/steps/s-IVXTXADABLBE/stderr.gz ⓘ Details: org.apache.hadoop.mapred.FileAlreadyExistsException: Output directory s3://bucket/output already exists JAR location: s3://bucket/hadoop-mapreduce-examples-2.7.2-amzn-1.jar Main class: None Arguments: wordcount s3://bucket/input/hello.txt s3://bucket/output Action on failure: Continue			
●	● s-3R6M87MQTDGPS	WordCount	Failed
●	● s-43HMPMI46PES7	Custom JAR	Failed
●	s-21EQLTWVIUTUV	Setup hadoop debugging	Completed

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-1K48XXXXXXHCB --step-id s-3QM0XXXXXM1W
```

The output will look similar to the following:

```
{
  "Step": {
    "Status": {
      "FailureDetails": {
        "LogFile": "s3://myBucket/logs/j-1K48XXXXXXHCB/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 Application History

Using Amazon EMR version 5.8.0 or later, you can view YARN application details using the **Application history** tab of a cluster's detail page in the console. Using Amazon EMR application history makes it easier for you to troubleshoot and analyze active jobs and job history. Instead of setting up and connecting to the master node to view open-source troubleshooting UIs or sift through log files, you can quickly view application metrics and access relevant log files.

Application history is enabled automatically for all clusters that run YARN applications. No special setup is required. Amazon EMR keeps historical information for up to seven days after an application has completed. Detailed application history is available only for Spark. There are additional minor limitations in Amazon EMR version 5.8.0. For more information, see [Known Issues](#) for version 5.8.0 in the *Amazon EMR Release Guide*.

Example: View Job Details for a Spark Application

The following sequence demonstrates a drill-down through a Spark application into job detail to evaluate stages, tasks, and executors using the **Application history** on a cluster details page (to view cluster details, from the **Clusters** list, select a cluster **Name**).

On the **Application history** tab, two rows are expanded to show the diagnostic summaries for two different Spark applications, and then an **Application ID** is selected to view further application detail:

Application ID	Type	Action	Status	Start time (UTC-7)	Duration	Finish time (UTC-7)	User
▶ application_1505786029486_0006	Spark	spark-wordcount.py	Failed	2017-09-18 18:58 (UTC-7)	22 s	2017-09-18 18:58 (UTC-7)	hadoop
▶ application_1505786029486_0005	Spark	spark-wordcount.py	Failed	2017-09-18 18:57 (UTC-7)	23 s	2017-09-18 18:58 (UTC-7)	hadoop
▶ application_1505786029486_0004	Spark	spark-wordcount.py	Failed	2017-09-18 18:57 (UTC-7)	22 s	2017-09-18 18:57 (UTC-7)	hadoop
▶ application_1505786029486_0002	Spark	spark-wordcount.py	Failed	2017-09-18 18:57 (UTC-7)	23 s	2017-09-18 18:57 (UTC-7)	hadoop
Diagnostics: User application exited with status 1							
▶ application_1505786029486_0001							
Spark							
spark-wordcount.py							
Diagnostics: Succeeded							

On the **Jobs** tab of **YARN application** details, the **Description of Job 0** is selected to see Job 0 details:

Job ID	Status	Description	Submitted (UTC-7)	Duration	Stages succeeded / total	Tasks succeeded / total
0	Succeeded	saveAsTextFile at NativeMethodAccessorsImpl.java:0	2017-09-18 18:57 (UTC-7)	15 s	2 / 2	4 / 4

On the **Job 0** details page, information about individual job stages is expanded, and then the **Description** for Stage 1 is selected to see Stage 1 details:

Amazon EMR Management Guide

View Log Files

Cluster: Holmes **Waiting** Cluster ready after last step failed.

[Summary](#) [Application history](#) [Monitoring](#) [Hardware](#) [Events](#) [Steps](#) [Configurations](#) [Bootstrap actions](#)

[Application history > application_1505786029486_0001 \(Spark\) C](#)

[Jobs](#) [Stages](#) [Executors](#)

Jobs > Job 0

Status: Succeeded
Completed stages: 2

Stages (2)

Stage ID	Status	Description	Submitted (UTC-7)	Duration	Tasks succeeded / total	Input	Output	Shuffle read	Shuffle write
1	Complete	saveAsTextFile at NativeMethodAccessormpl.java:0	2017-09-18 18:57 (UTC-7)	0.5 s	2 / 2	1.4 kB	2.0 kB		

Details: org.apache.spark.api.java.AbstractJavaRDDLike saveAsTextFile(JavaRDDLike scala:45)
sun.reflect.NativeMethodAccessorImpl invoke0(Native Method)
sun.reflect.NativeMethodAccessorImpl invoke(NativeMethodAccessormpl.java:62)
sun.reflect.DelegatingMethodAccessorImpl invoke(DelegatingMethodAccessormpl.java:43)
java.lang.reflect.Method invoke(Method.java:498)
java.lang.reflect.Method.invoke(MethodInvoker.java:244)
py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:257)
py4j.Gateway invoke(Gateway.java:280)
py4j.commands.AbstractCommand invokeMethod(AbstractCommand.java:132)
py4j.commands.CallCommand execute(CallCommand.java:79)
py4j.GatewayConnection run(GatewayConnection.java:214)
java.lang.Thread run(Thread.java:748)

▶ 0 Complete reduceByKey at spark-wordcount.py:10 2017-09-18 18:57 (UTC-7) 6 s 2 / 2 1.6 kB 2.0 kB

On the **Stage 1** details page, key metrics for stage tasks and executors can be seen, and task and executor logs can be viewed using links:

[Summary](#) [Application history](#) [Monitoring](#) [Hardware](#) [Events](#) [Steps](#) [Configurations](#) [Bootstrap actions](#)

[Application history > application_1505786029486_0001 \(Spark\) C](#)

[Jobs](#) [Stages](#) [Executors](#)

Jobs > Job 0 > Stage 1 (attempt 0)

Total time across all tasks: 0.7 s
Locality level summary: Node local: 2
Output (size / records): 1.4 kB / 101
Shuffle read (size / records): 2.0 kB / 12

Summary metrics for 2 completed tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	0.4 s	0.4 s	0.4 s	0.4 s	0.4 s
GC time	8 ms	8 ms	8 ms	8 ms	8 ms
Output (size / records)	680.0 B / 46	680.0 B / 46	779.0 B / 55	779.0 B / 55	779.0 B / 55
Result serialization time	1 ms	1 ms	1 ms	1 ms	1 ms
Shuffle read (size / records)	914.0 B / 6	914.0 B / 6	1.1 kB / 6	1.1 kB / 6	1.1 kB / 6
Shuffle remote reads	0.0 B	0.0 B	0.0 B	0.0 B	0.0 B
Task deserialization time	64 ms	64 ms	65 ms	65 ms	65 ms

Aggregated metrics by executor (1)

Executor ID	Address	Task time	Total tasks	Failed tasks	Succeeded tasks	Output	Shuffle read	Blacklisted
1	ip-10-218-185-134.ec2.internal:34689	0.9 s	4	0	2	1.4 kB	2.0 kB	

View logs (circled)

Tasks (2)

ID	Attempt	Status	Locality level	Executor ID / Host	Launch time (UTC-7)	Duration	Task deserialization time	GC time	Result serialization time	Output size / records	Shuffle read size / records	Shuffle remote reads	Errors
2	0	Succeeded	Node local	1 / ip-10-218-185-134.ec2.internal	2017-09-18 18:57 (UTC-7)	0.4 s	64 ms	8 ms	1 ms	779.0 B / 55	1.1 kB / 6	0.0 B	
3	0	Succeeded	Node local	1 / ip-10-218-185-134.ec2.internal	2017-09-18 18:57 (UTC-7)	0.4 s	65 ms	8 ms	1 ms	680.0 B / 46	914.0 B / 6	0.0 B	

View logs (circled)

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 \(p. 212\)](#)
- [View Log Files Archived to Amazon S3 \(p. 213\)](#)
- [View Log Files in the Debugging Tool \(p. 214\)](#)

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 (Amazon EMR 4.6.0 and earlier)	Instance controller logs.
/emr/instance-controller (Amazon EMR 4.7.0 and later)	
/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 (Amazon EMR 4.6.0 and earlier)	Logs written by the service nanny process.
/emr/service-nanny (Amazon EMR 4.7.0 and later)	
/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>	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-1234ABCDEGHI and s-5678IJKLMNOP. The first step is located in /mnt/var/log/hadoop/steps/s-1234ABCDEGHI/ and the second step in /mnt/var/log/hadoop/steps/s-5678IJKLMNOP/. The step logs written by Amazon EMR are as follows. <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 \(p. 239\)](#).
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 file viewer of your choice to view the log file. The following example uses the Linux less command to view the controller log file.

```
less 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: s-1234ABCDEFGH and s-5678IJKLMNOP. 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.

Location	Description
	<ul style="list-style-type: none"> • 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.
/ <i>JobFlowId</i> /containers	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

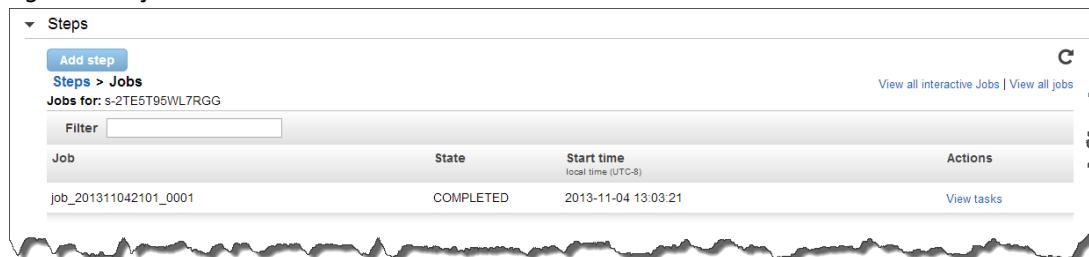
Amazon EMR does not automatically enable the debugging tool. You must configure this when you launch the cluster.

To view cluster logs using the console

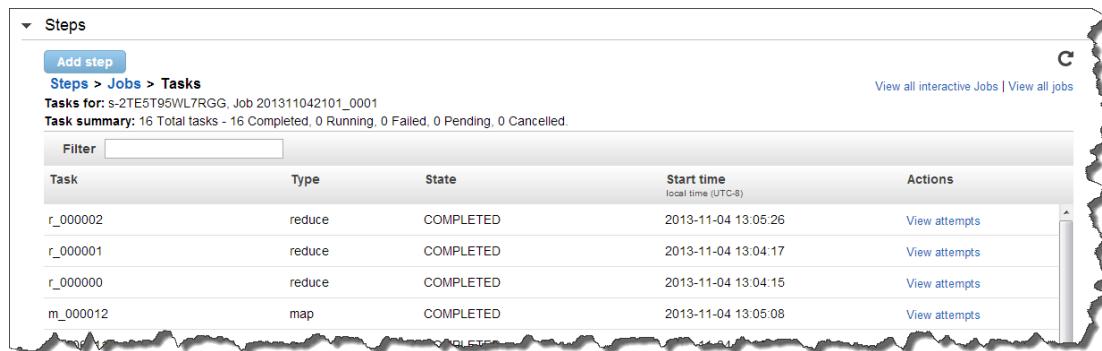
1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. From the **Cluster List** page, choose the details icon next to the cluster you want to view.

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.



6. To view the logs generated by a task attempt, choose the **stderr**, **stdout**, and **syslog** links to the right of the task attempt.



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 \(p. 216\)](#)
- [Monitor Metrics with CloudWatch \(p. 223\)](#)

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 <i>ClusterID</i> (<i>ClusterName</i>) was requested at <i>Time</i> 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 <i>ClusterID</i> (<i>ClusterName</i>) is being created in subnet (<i>SubnetName</i>) in VPC (<i>VPCName</i>) in availability zone (<i>AvailabilityZoneID</i>), 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 <i>ClusterID</i> (<i>ClusterName</i>) is being created in availability zone (<i>AvailabilityZoneID</i>), which was chosen from the specified availability zone options.</p>
RUNNING	INFO	Amazon EMR cluster <i>ClusterID</i> (<i>ClusterName</i>) began running steps at <i>Time</i> .
WAITING	INFO	<p>Amazon EMR cluster <i>ClusterID</i> (<i>ClusterName</i>) was created at <i>Time</i> and is ready for use.</p> <p>—or—</p> <p>Amazon EMR cluster <i>ClusterID</i> (<i>ClusterName</i>) finished running all pending steps at <i>Time</i>.</p> <p>Note A cluster in the WAITING state may nevertheless be processing jobs.</p>

State or State Change	Severity	Message
TERMINATED	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: INTERNAL_ERROR, VALIDATION_ERROR, INSTANCE_FAILURE, BOOTSTRAP_FAILURE, or STEP_FAILURE. • INFO if the cluster terminated with any of the following state change reasons: USER_REQUEST or ALL_STEPS_COMPLETED. 	Amazon EMR Cluster <i>ClusterId</i> (<i>ClusterName</i>) has terminated at <i>Time</i> with a reason of <i>StateChangeReason:Code</i> .
TERMINATED_WITH_ERRORS	CRITICAL	Amazon EMR Cluster <i>ClusterId</i> (<i>ClusterName</i>) has terminated with errors at <i>Time</i> with a reason of <i>StateChangeReason:Code</i> .

Instance Fleet Events

Note

The instance fleets configuration is available only in Amazon EMR release versions 4.8.0 and later, excluding 5.0.0 and 5.0.3.

State or State Change	Severity	Message
From PROVISIONING to WAITING	INFO	Provisioning for instance fleet <i>InstanceFleetID</i> in Amazon EMR cluster <i>ClusterId</i> (<i>ClusterName</i>) is complete. Provisioning started at <i>Time</i> and took <i>Num</i> minutes. The instance fleet now has On-Demand capacity of <i>Num</i> and Spot capacity of <i>Num</i> . Target On-Demand capacity was <i>Num</i> , and target Spot capacity was <i>Num</i> .
From WAITING to RESIZING	INFO	A resize for instance fleet <i>InstanceFleetID</i> in Amazon EMR cluster <i>ClusterId</i> (<i>ClusterName</i>) started at <i>Time</i> . The instance fleet is resizing from an On-Demand capacity of <i>Num</i> to a target of <i>Num</i> , and from a Spot capacity of <i>Num</i> to a target of <i>Num</i> .

State or State Change	Severity	Message
From RESIZING to WAITING	INFO	The resizing operation for instance fleet <i>InstanceFleetID</i> in Amazon EMR cluster <i>ClusterId</i> (<i>ClusterName</i>) is complete. The resize started at <i>Time</i> and took <i>Num</i> minutes. The instance fleet now has On-Demand capacity of <i>Num</i> and Spot capacity of <i>Num</i> . Target On-Demand capacity was <i>Num</i> and target Spot capacity was <i>Num</i> .
From RESIZING to WAITING	WARN	The resizing operation for instance fleet <i>InstanceFleetID</i> in Amazon EMR cluster <i>ClusterId</i> (<i>ClusterName</i>) has reached the timeout and stopped. The resize started at <i>Time</i> and stopped after <i>Num</i> minutes. The instance fleet now has On-Demand capacity of <i>Num</i> and Spot capacity of <i>Num</i> . Target On-Demand capacity was <i>Num</i> and target Spot capacity was <i>Num</i> .
ARRESTED	ERROR	Instance fleet <i>InstanceFleetID</i> in Amazon EMR cluster <i>ClusterId</i> (<i>ClusterName</i>) was arrested at <i>Time</i> for the following reason: <i>ReasonDesc</i> .
RESIZING	WARNING	The resizing operation for instance fleet <i>InstanceFleetID</i> in Amazon EMR cluster <i>ClusterId</i> (<i>ClusterName</i>) is stuck for the following reason: <i>ReasonDesc</i> .
WAITING OR RUNNING	INFO	A resize for instance fleet <i>InstanceFleetID</i> in Amazon EMR cluster <i>ClusterId</i> (<i>ClusterName</i>) was initiated by <i>Entity</i> at <i>Time</i> .

Instance Group Events

State or State Change	Severity	Message
From RESIZING to RUNNING	INFO	The resizing operation for instance group <i>InstanceGroupId</i> in Amazon

State or State Change	Severity	Message
		EMR cluster <i>ClusterId</i> (<i>ClusterName</i>) is complete. It now has an instance count of <i>Num</i> . The resize started at <i>Time</i> and took <i>Num</i> minutes to complete.
From RUNNING to RESIZING	INFO	A resize for instance group <i>InstanceGroupID</i> in Amazon EMR cluster <i>ClusterId</i> (<i>ClusterName</i>) started at <i>Time</i> . It is resizing from an instance count of <i>Num</i> to <i>Num</i> .
ARRESTED	ERROR	Instance group <i>InstanceGroupID</i> in Amazon EMR cluster <i>ClusterId</i> (<i>ClusterName</i>) was arrested at <i>Time</i> for the following reason: <i>ReasonDesc</i> .
RESIZING	WARNING	The resizing operation for instance group <i>InstanceGroupID</i> in Amazon EMR cluster <i>ClusterId</i> (<i>ClusterName</i>) is stuck for the following reason: <i>ReasonDesc</i> .
WAITING or RUNNING	INFO	A resize for instance group <i>InstanceGroupID</i> in Amazon EMR cluster <i>ClusterId</i> (<i>ClusterName</i>) was initiated by <i>Entity</i> at <i>Time</i> .

Automatic Scaling Policy Events

State or State Change	Severity	Message
PENDING	INFO	An Auto Scaling policy was added to instance group <i>InstanceGroupID</i> in Amazon EMR cluster <i>ClusterId</i> (<i>ClusterName</i>) at <i>Time</i> . The policy is pending attachment. —or— The Auto Scaling policy for instance group <i>InstanceGroupID</i> in Amazon EMR cluster <i>ClusterId</i> (<i>ClusterName</i>) was updated at <i>Time</i> . The policy is pending attachment.

State or State Change	Severity	Message
ATTACHED	INFO	The Auto Scaling policy for instance group <i>InstanceGroupId</i> in Amazon EMR cluster <i>ClusterId</i> (<i>ClusterName</i>) was attached at <i>Time</i> .
DETACHED	INFO	The Auto Scaling policy for instance group <i>InstanceGroupId</i> in Amazon EMR cluster <i>ClusterId</i> (<i>ClusterName</i>) was detached at <i>Time</i> .
FAILED	ERROR	<p>The Auto Scaling policy for instance group <i>InstanceGroupId</i> in Amazon EMR cluster <i>ClusterId</i> (<i>ClusterName</i>) could not attach and failed at <i>Time</i>.</p> <p>—or—</p> <p>The Auto Scaling policy for instance group <i>InstanceGroupId</i> in Amazon EMR cluster <i>ClusterId</i> (<i>ClusterName</i>) could not detach and failed at <i>Time</i>.</p>

Step Events

State or State Change	Severity	Message
PENDING	INFO	Step <i>StepID</i> (<i>StepName</i>) was added to Amazon EMR cluster <i>ClusterId</i> (<i>ClusterName</i>) at <i>Time</i> and is pending execution.
CANCEL_PENDING	WARN	Step <i>StepID</i> (<i>StepName</i>) in Amazon EMR cluster <i>ClusterId</i> (<i>ClusterName</i>) was cancelled at <i>Time</i> and is pending cancellation.
RUNNING	INFO	Step <i>StepID</i> (<i>StepName</i>) in Amazon EMR cluster <i>ClusterId</i> (<i>ClusterName</i>) started running at <i>Time</i> .
COMPLETED	INFO	Step <i>StepID</i> (<i>StepName</i>) in Amazon EMR cluster <i>ClusterId</i> (<i>ClusterName</i>) completed execution at <i>Time</i> . The step

State or State Change	Severity	Message
		started running at <i>Time</i> and took <i>Num</i> minutes to complete.
CANCELLED	WARN	Cancellation request has succeeded for cluster step <i>StepID</i> (<i>StepName</i>) in Amazon EMR cluster <i>ClusterId</i> (<i>ClusterName</i>) at <i>Time</i> , and the step is now cancelled.
FAILED	ERROR	Step <i>StepID</i> (<i>StepName</i>) in Amazon EMR cluster <i>ClusterId</i> (<i>ClusterName</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 with the 'JeffGollAutoScale' cluster selected. The 'Events' section is expanded, showing two log entries:

- Dec 14 02:15 PM: A resize for your Amazon EMR cluster j-3CHYFADDOEDEM (JeffGollAutoScale) is complete and instance group i-140CC0B9592TIZ is VANTING has an instance count of 1. The resize was initiated at 2016-12-14 20:34 UTC and took 101 minutes to complete.
- Dec 14 12:35 PM: A resize for your Amazon EMR cluster j-3CHYFADDOEDEM (JeffGollAutoScale) was started at 2016-12-14 20:34 UTC. Instance group i-140CC0B9592TIZ RESIZED from instance count of 2 to 1.

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 \(p. 226\)](#).

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.

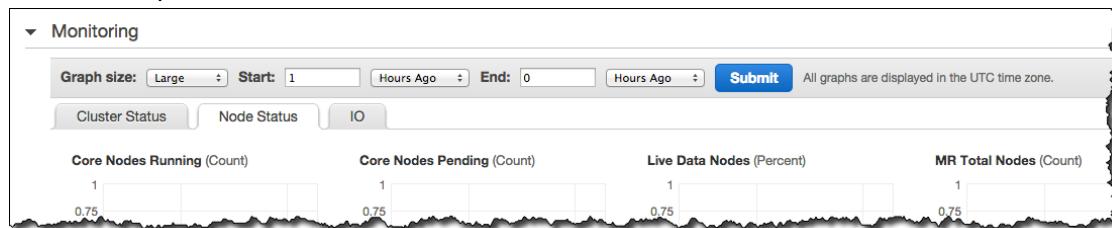
Accessing CloudWatch Metrics

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.

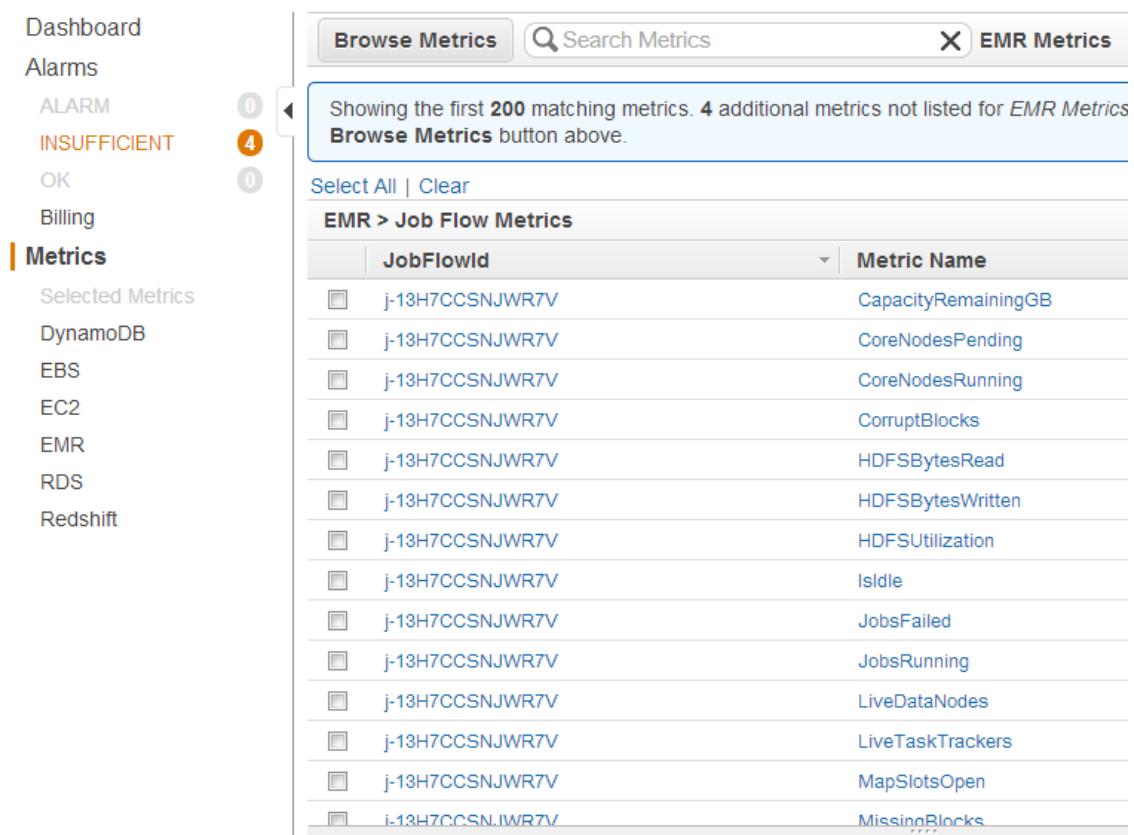
To view metrics in the Amazon EMR console

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.

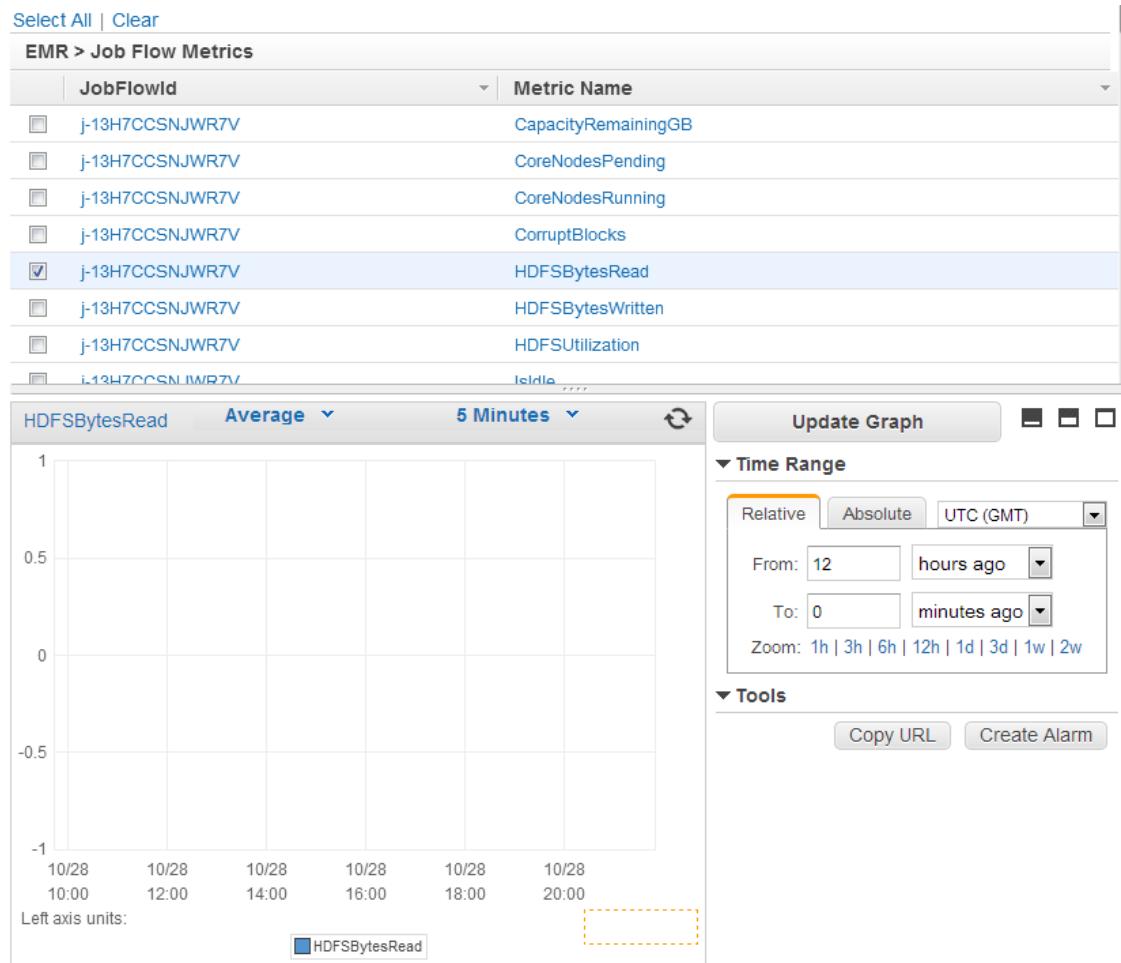


To view metrics in the CloudWatch console

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.



4. Open a metric to display the graph.



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 tables list 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 metrics are available for clusters running Hadoop 2.x versions.

Metric	Description
<i>Cluster Status</i>	
IsIdle	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

Metric	Description
	<p>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> <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 ContainerPendingRatio represents a number, not a percentage. This value is useful for scaling cluster resources based on container allocation behavior.</p> <p>Units: <i>Count</i></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>

Metric	Description
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>
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. Equivalent to YARN metric <code>mapred.resourcemanager.TotalNodes</code>.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>

Metric	Description
MRActiveNodes	<p>The number of nodes presently running MapReduce tasks or jobs. Equivalent to YARN metric <code>mapred.resourcemanager.NoOfActiveNodes</code>.</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. Equivalent to YARN metric <code>mapred.resourcemanager.NoOfLostNodes</code>.</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. Equivalent to YARN metric <code>mapred.resourcemanager.NoOfUnhealthyNodes</code>.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
MRDecommissionedNodes	<p>The number of nodes allocated to MapReduce applications that have been marked in a DECOMMISSIONED state. Equivalent to YARN metric <code>mapred.resourcemanager.NoOfDecommissionedNodes</code>.</p> <p>Use case: Monitor cluster health, Monitor cluster progress</p> <p>Units: <i>Count</i></p>
MRRebootedNodes	<p>The number of nodes available to MapReduce that have been rebooted and marked in a REBOOTED state. Equivalent to YARN metric <code>mapred.resourcemanager.NoOfRebootedNodes</code>.</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>Count</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>Count</i></p>

Metric	Description
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. This metric aggregates MapReduce jobs only, and does not apply for other workloads on EMR.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: <i>Count</i></p>
HDFSBytesWritten	<p>The number of bytes written to HDFS. This metric aggregates MapReduce jobs only, and does not apply for other workloads on EMR.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: <i>Count</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>
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>

Metric	Description
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>
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>

Metric	Description
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 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>

Metric	Description
	Units: <i>Count</i>
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> <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>

Metric	Description
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>
TaskNodesPending	<p>The number of task 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. This metric aggregates MapReduce jobs only, and does not apply for other workloads on EMR.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: <i>Count</i></p>
S3BytesRead	<p>The number of bytes read from Amazon S3. This metric aggregates MapReduce jobs only, and does not apply for other workloads on EMR.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: <i>Count</i></p>

Metric	Description
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>
TotalLoad	<p>The current, total number of readers and writers reported by all DataNodes in a cluster.</p> <p>Use case: Diagnose the degree to which high I/O might be contributing to poor job execution performance. Worker nodes running the DataNode daemon must also perform map and reduce tasks. Persistently high TotalLoad values over time can indicate that high I/O might be a contributing factor to poor performance. Occasional spikes in this value are typical and do not usually indicate a problem.</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>

Metric	Description
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-xxxxxxxxxxxxxx. 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_XXXXXXXXXXXX_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 [Ganglia](#) in the *Amazon EMR Release Guide*.

Logging Amazon EMR API Calls in AWS CloudTrail

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

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

Amazon EMR Information in CloudTrail

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

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

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

All Amazon EMR actions are logged by CloudTrail and are documented in the [Amazon EMR API Reference](#). For example, calls to the `RunJobFlow`, `ListCluster` and `DescribeCluster` actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

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

Example: Amazon EMR Log File Entries

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

The following example shows a CloudTrail log entry that demonstrates the `RunJobFlow` action.

```
{  
  "Records": [  
    {  
      "eventVersion": "1.01",  
      "userIdentity": {  
        "type": "IAMUser",  
        "principalId": "EX_PRINCIPAL_ID",  
        "arn": "arn:aws:iam::123456789012:user/temporary-user-xx-7M",  
        "accountId": "123456789012",  
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",  
        "sessionContext": {  
          "attributes": {},  
          "createDate": "2014-01-01T12:00:00Z",  
          "expiration": "2014-01-01T12:05:00Z",  
          "sessionIssuer": {  
            "arn": "arn:aws:iam::123456789012:root",  
            "type": "AWSAccountRoot"  
          },  
          "sessionId": "AIEEXAMPLESESSIONID",  
          "userName": "temporary-user-xx-7M"  
        }  
      }  
    }  
  ]  
}
```

```

        "accountId":"123456789012",
        "userName":"temporary-user-xx-7M"
    },
    "eventTime":"2018-03-31T17:59:21Z",
    "eventSource":"elasticmapreduce.amazonaws.com",
    "eventName":"RunJobFlow",
    "awsRegion":"us-west-2",
    "sourceIPAddress":"192.0.2.1",
    "userAgent":"aws-sdk-java/unknown-version Linux/xx Java_HotSpot(TM)_64-
Bit_Server_VM/xx",
    "requestParameters":{
        "tags": [
            {
                "value": "prod",
                "key": "domain"
            },
            {
                "value": "us-west-2",
                "key": "realm"
            },
            {
                "value": "VERIFICATION",
                "key": "executionType"
            }
        ],
        "instances": {
            "slaveInstanceType": "m4.large",
            "ec2KeyName": "emr-integtest",
            "instanceCount": 1,
            "masterInstanceType": "m4.large",
            "keepJobFlowAliveWhenNoSteps": true,
            "terminationProtected": false
        },
        "visibleToAllUsers": false,
        "name": "MyCluster",
        "ReleaseLabel": "emr-5.16.0"
    },
    "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 the master node, and for core and task nodes, that determine how you access the nodes.

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 authenticate to the cluster. You can either use Kerberos for authentication, or specify an Amazon EC2 key pair private key when you launch the cluster. For more information about configuring Kerberos, and then connecting, see [Use Kerberos Authentication \(p. 144\)](#). When 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 \(p. 239\)](#)
- [View Web Interfaces Hosted on Amazon EMR Clusters \(p. 243\)](#)

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 \(p. 243\)](#).

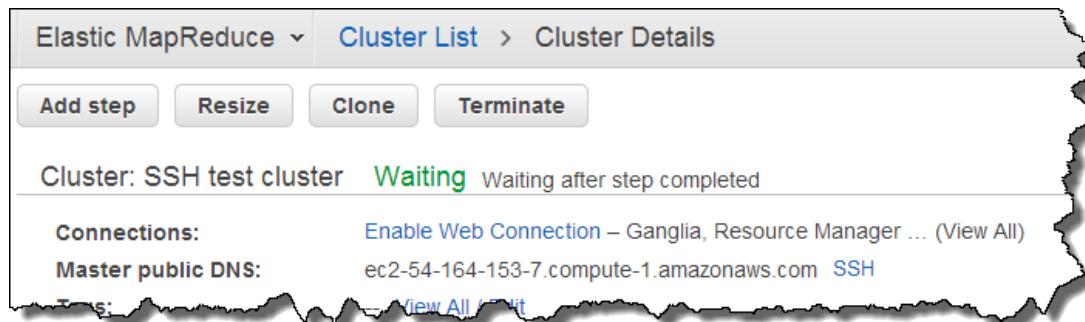
To connect to the master node using SSH, you need the public DNS name of the master node.

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.

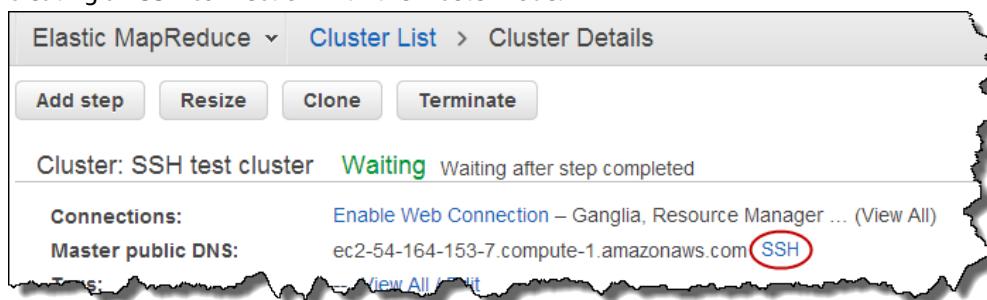
To retrieve the public DNS name of the master node using the Amazon EMR console

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-12XXXXXXXXXFMH",  
"PrivateIpAddress": "##.##.##.##"
```

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

Connect to the Master Node Using SSH and an Amazon EC2 Private Key on Linux, Unix, and Mac OS X

To create an SSH connection authenticated with a private key file, you need to specify the Amazon EC2 key pair private key when you launch a 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*.

Your Linux computer most likely includes an SSH client by default. For example, OpenSSH is installed on most Linux, Unix, and macOS operating systems. You can check for an SSH client by typing `ssh` at the command line. If your computer does not recognize the command, 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 \(p. 241\)](#).

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, when using EMR-Managed Security Groups, these web sites are only available on the master node's local web server, so you need to connect to the master node to view them. For more information, see [Connect to the Master Node Using SSH \(p. 239\)](#). Hadoop also publishes user interfaces as web sites hosted on the core and task nodes. These web sites are also only available on local web servers on the nodes.

Warning

It is possible to configure a custom security group to allow inbound access to these web interfaces. Keep in mind that any port on which you allow inbound traffic represents a potential security vulnerability. Carefully review custom security groups to ensure that you minimize vulnerabilities. For more information, see [Control Network Traffic with Security Groups \(p. 168\)](#).

The following table lists web interfaces that you can view on cluster instances. These Hadoop interfaces are available on all clusters. For the master instance interfaces, replace `master-public-dns-name` with the **Master public DNS** listed on the cluster **Summary** tab in the EMR console. For core and task instance interfaces, replace `coretask-public-dns-name` with the **Public DNS name** listed for the instance. To find an instance's **Public DNS name**, in the EMR console, choose your cluster from the list, choose the **Hardware** tab, choose the **ID** of the instance group that contains the instance you want to connect to, and then note the **Public DNS name** listed for the instance.

Important

To access web interfaces, you must edit the security groups associated with master and core instances so that they have an inbound rule that allows SSH traffic (port 22) from trusted clients, such as your computer's IP address. 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://coretask-public-dns-name:8042/</code>
Hadoop HDFS NameNode	<code>http://master-public-dns-name:50070/</code>
Hadoop HDFS DataNode	<code>http://coretask-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 \(p. 248\)](#).

Topics

- [Option 1: Set Up an SSH Tunnel to the Master Node Using Local Port Forwarding \(p. 245\)](#)
- [Option 2, Part 1: Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding \(p. 246\)](#)
- [Option 2, Part 2: Configure Proxy Settings to View Websites Hosted on the Master Node \(p. 248\)](#)
- [Access the Web Interfaces on the Master Node Using the Console \(p. 250\)](#)

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 \(p. 239\)](#). 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 \(p. 243\)](#).

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 8088 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 \(p. 239\)](#). 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 \(p. 243\)](#).

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@c2-###-##-##-##.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 \(p. 248\)](#).
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 \(p. 241\)](#). 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 \(p. 242\)](#).
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 \(p. 248\)](#).
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 <https://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 \(p. 248\)](#).
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 \(p. 246\)](#). For more information about the available web interfaces, see [View Web Interfaces Hosted on Amazon EMR Clusters \(p. 243\)](#).

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** in the example. You should set this port 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 \(p. 246\)](#). 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" />
      </matches>
    </proxy>
  </proxies>
</foxyproxy>
```

```
<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.

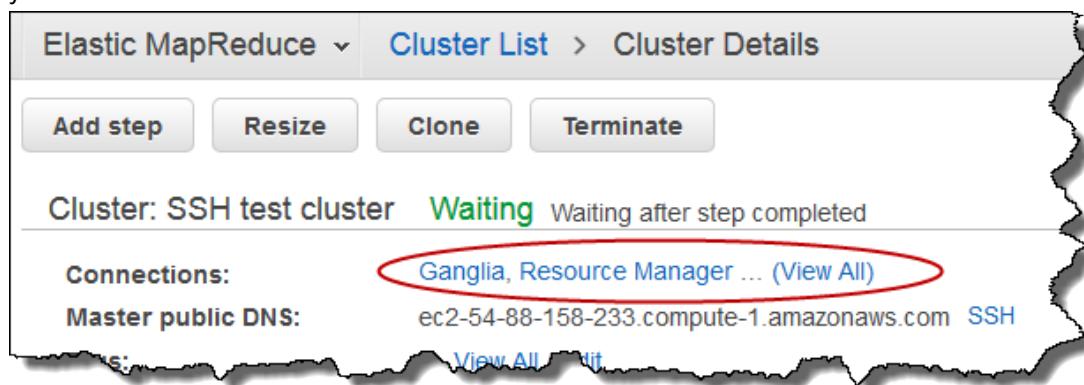
For a complete list of web interfaces on the master node, see [View Web Interfaces Hosted on Amazon EMR Clusters \(p. 243\)](#).

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, these websites 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 EC2 URLs.

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.

Elastic MapReduce ▾ Cluster List > Cluster Details

Add step Resize Clone Terminate

Cluster: SSH test cluster Waiting Waiting after step completed

Connections: [Enable Web Connection](#) (Ganglia, Resource Manager ... (View All))

Master public DNS: ec2-54-88-158-233.compute-1.amazonaws.com [SSH](#)

Tags: – [View All / Edit](#)

Note

If you have an SSH tunnel configured using local port forwarding, the Amazon EMR console does not detect the connection.

Terminate a Cluster

This section describes the methods of terminating a cluster. For information about enabling termination protection and auto-terminating clusters, see [Control Cluster Termination \(p. 78\)](#). 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 disable termination protection before 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-3KVXXXXXXX7UG` with your cluster ID.

```
aws emr terminate-clusters --cluster-ids j-3KVXXXXXXX7UG
```

To terminate multiple clusters, type the following command and replace `j-3KVXXXXXXX7UG` and `j-WJ2XXXXXXX8EU` with your cluster IDs.

```
aws emr terminate-clusters --cluster-ids j-3KVXXXXXXX7UG j-WJ2XXXXXXX8EU
```

For more information on using Amazon EMR commands in the AWS CLI, see <https://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.

1. Type the following command to disable termination protection and replace `j-3KVTXXXXXX7UG` with your cluster ID.

```
aws emr modify-cluster-attributes --cluster-id j-3KVTXXXXXX7UG --no-termination-protected
```

2. To terminate the cluster, type the following command and replace `j-3KVXXXXXXX7UG` with your cluster ID.

```
aws emr terminate-clusters --cluster-ids j-3KVXXXXXXX7UG
```

To terminate multiple clusters, type the following command and replace `j-3KVXXXXXXX7UG` and `j-WJ2XXXXXX8EU` with your cluster IDs.

```
aws emr terminate-clusters --cluster-ids j-3KVXXXXXXX7UG j-WJ2XXXXXX8EU
```

For more information on using Amazon EMR commands in the AWS CLI, see <https://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](#).

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:

- Using Amazon EMR versions 4.x and later, you can configure *automatic scaling* for the core instance group and task instance groups when you first create them or after the cluster is 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. When using Amazon EMR version 5.1.0 or later, 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 [Cluster Scale-Down \(p. 268\)](#).

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 \(p. 254\)](#)
- [Manually Resizing a Running Cluster \(p. 262\)](#)
- [Cluster Scale-Down \(p. 268\)](#)

Using Automatic Scaling in Amazon EMR

Automatic scaling in Amazon EMR release versions 4.0 and later allows you to programmatically scale out and scale in core nodes and task nodes based on a CloudWatch metric and other parameters that you specify in a *scaling policy*. Automatic scaling is available with the instance groups configuration and is not available when you use instance fleets. For more information about instance groups and instance fleets, see [Create a Cluster with Instance Fleets or Uniform Instance Groups \(p. 111\)](#).

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. In addition, when using the AWS CLI or the Amazon EMR API, you can specify custom CloudWatch metrics. Custom metrics are not available for selection using the AWS Management Console. 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 configured with the appropriate role

policy and trust policy, `EMR_AutoScaling_DefaultRole`, is available for this purpose. When you create a cluster with a scaling policy for the first time using the AWS Management Console, Amazon EMR creates the default role and attaches the default managed policy for permissions, `AmazonElasticMapReduceforAutoScalingRole`.

When you create a cluster with an automatic scaling policy using the AWS CLI, you must first ensure that either the default IAM role exists, or that you have a custom IAM role with a policy attached that provides the appropriate permissions. To create the default role, you can run the `create-default-roles` command before you create a cluster. You can then specify `--auto-scaling-role EMR_AutoScaling_DefaultRole` option when you create a cluster. Alternatively, you can create a custom automatic scaling role and then specify it when you create a cluster, for example `--auto-scaling-role MyEMRAutoScalingRole`. If you create a customized automatic scaling role for Amazon EMR, we recommend that you base permissions policies for your custom role based on the managed policy. For more information, see [Configure IAM Roles for Amazon EMR Permissions to AWS Services \(p. 187\)](#).

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 *Amazon EC2 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 [Cluster Scale-Down \(p. 268\)](#).

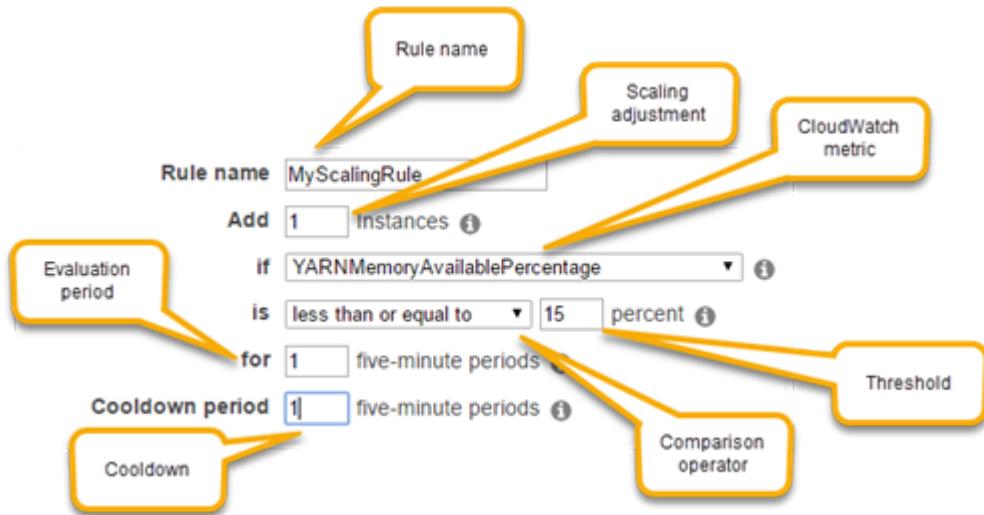
The following parameters for each rule in a policy determine automatic scaling behavior.

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**, in five-minute increments, 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 *Amazon EC2 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—

2. 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` `MyAutoScalingRole` command to specify the IAM role for autoscaling. The default role is `EMR_AutoScaling_DefaultRole` and can be created with the `create-default-roles` command. 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 automatic scaling policy using the AWS Management Console for Amazon EMR. For brevity, a scale-in policy is not shown. We do not recommend creating a scale-out rule without a scale-in rule.

```
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=m4.large,InstanceCount=1 'Name=MyCoreIG,InstanceGroupType=CORE,InstanceType=m4.large,InstanceCount=2,AutoScalingPolicy={Constraints=[{MetricName=CPUUtilization,Threshold=80}],Description=Replicates the default scale-out rule in the console.,Action={SimpleScalingPolicyConfiguration={AdjustmentType=CHANGE_IN_CAPACITY,ScalingAdjustmentType=ChangeInCapacity,ElasticMapReduce,Period=300,Statistic=AVERAGE,Threshold=15,Unit=PERCENT,Dimensions=[{Key=JobFlowId,Value=MyJobFlow}],ScalingAdjustment=1},MetricName=CPUUtilization,MinAdjustment=1,MaxAdjustment=2,ScaleInCooldown=60,ScaleOutCooldown=60}}'
```

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": "m4.large"  
},  
{  
    "InstanceCount": 2,  
    "Name": "MyCoreIG",  
    "InstanceGroupType": "CORE",  
    "InstanceType": "m4.large",  
    "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,  
                                    "Comparison": "GreaterThanOrEqualTo",  
                                    "Value": 15  
                                }  
                            }  
                        }  
                    ]  
                }  
            }  
        ]  
    }  
}
```

```

        "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-1CWOHP4PI3OVJ"
                }
            ],
            "EvaluationPeriods": 1,
            "Period": 300,
            "ComparisonOperator": "GREATER_THAN",
            "Statistic": "AVERAGE"
        }
    },
    "Description": "",
    "Action": {
        "SimpleScalingPolicyConfiguration": {
            "CoolDown": 300,
            "AdjustmentType": "CHANGE_IN_CAPACITY",
            "ScalingAdjustment": -1
        }
    }
}
]
},
"Configurations": [],
"InstanceType": "m4.large",
"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": "m4.large",
  "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 add and remove instances from core and task instance groups and instance fleets in a running cluster using the AWS Management Console, AWS CLI, or the Amazon EMR API. If a cluster uses instance groups, you explicitly change the instance count. If your cluster uses instance fleets, you can change the target units for On-Demand Instances and Spot Instances. The instance fleet then adds and removes instances to meet the new target. For more information, see [Instance Fleet Options \(p. 112\)](#). Applications can use newly provisioned Amazon EC2 instances to host nodes as soon as the instances are available. When instances are removed, Amazon EMR terminates tasks in a way that does not interrupt jobs and safeguards against data loss. For more information, see [Terminate at Task Completion \(p. 268\)](#).

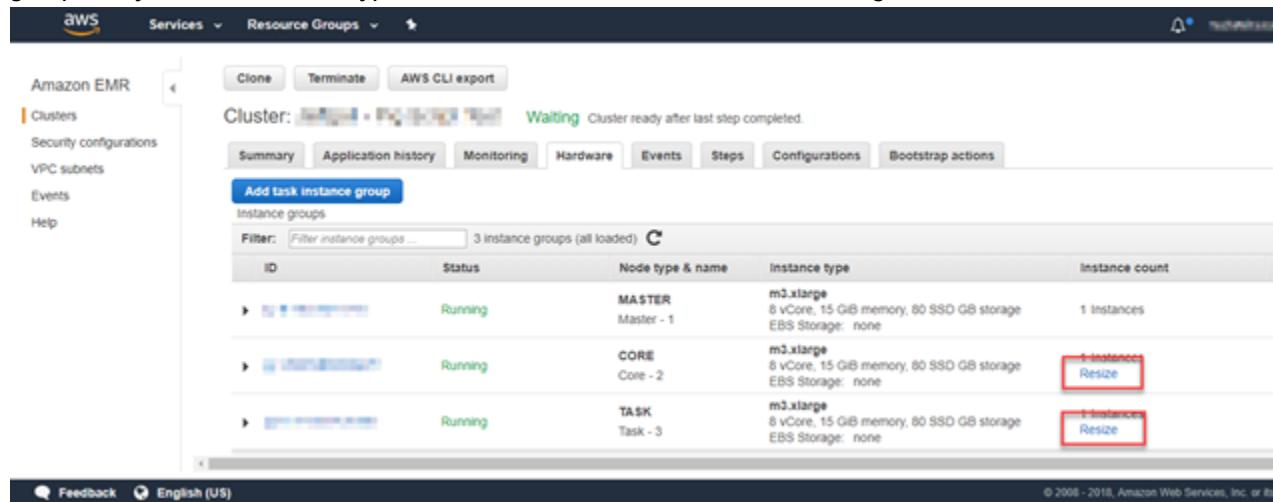
Resize a Cluster Using the Console

You can use the Amazon EMR console to resize a running cluster.

To change the instance count for an existing running cluster using the console

1. From the **Cluster List** page, choose a cluster to resize.
2. On the **Cluster Details** page, choose **Hardware**.

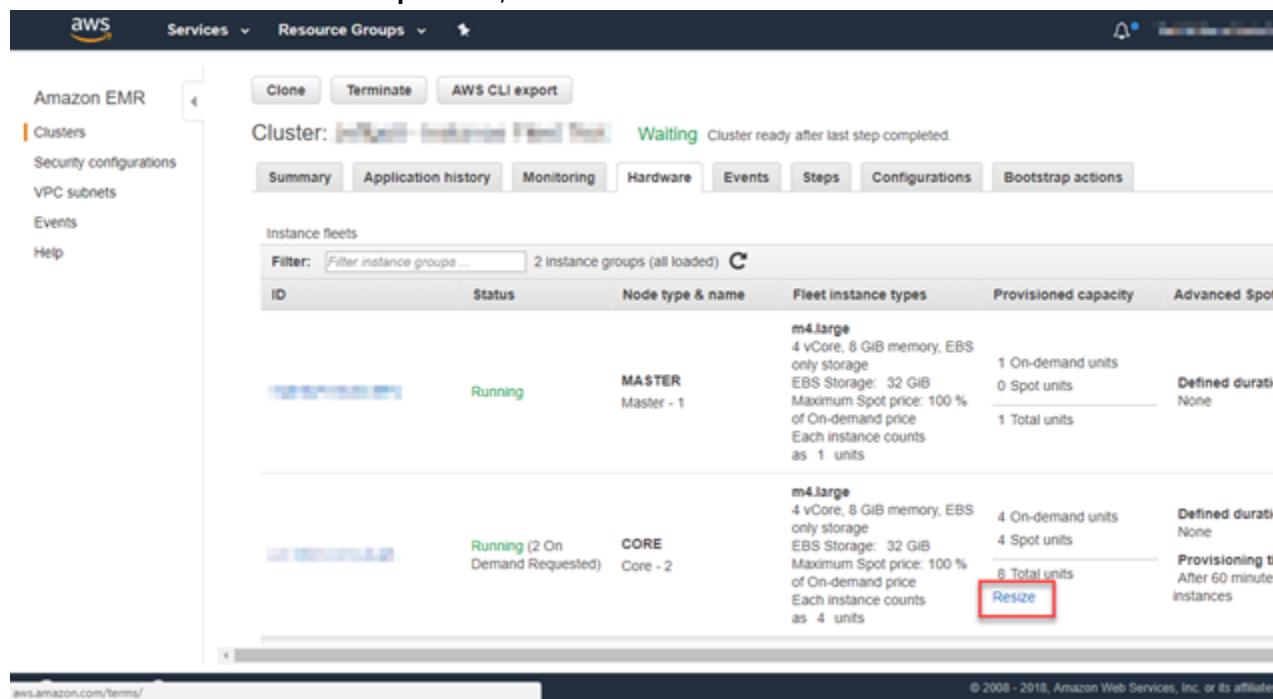
3. If your cluster uses instance groups, choose **Resize** in the **Instance count** column for the instance group that you want to resize, type a new instance count, and then select the green check mark.



ID	Status	Node type & name	Instance type	Instance count
MASTER	Running	MASTER Master - 1	m3.xlarge 8 vCore, 15 GiB memory, 80 SSD GB storage EBS Storage: none	1 Instances
CORE	Running	CORE Core - 2	m3.xlarge 8 vCore, 15 GiB memory, 80 SSD GB storage EBS Storage: none	1 Instances Resize
TASK	Running	TASK Task - 3	m3.xlarge 8 vCore, 15 GiB memory, 80 SSD GB storage EBS Storage: none	1 Instances Resize

-OR-

If your cluster uses instance fleets, choose **Resize** in the **Provisioned capacity** column, type new values for **On-demand units** and **Spot units**, and then choose **Resize**.



ID	Status	Node type & name	Fleet instance types	Provisioned capacity	Advanced Spot
MASTER	Running	MASTER Master - 1	m4.large 4 vCore, 8 GiB memory, EBS only storage EBS Storage: 32 GiB Maximum Spot price: 100 % of On-demand price Each instance counts as 1 units	1 On-demand units 0 Spot units <hr/> 1 Total units	Defined duration None
CORE	Running (2 On Demand Requested)	CORE Core - 2	m4.large 4 vCore, 8 GiB memory, EBS only storage EBS Storage: 32 GiB Maximum Spot price: 100 % of On-demand price Each instance counts as 4 units	4 On-demand units 4 Spot units <hr/> 8 Total units	Defined duration None <hr/> Provisioning time After 60 minutes Instances

When you make a change to the number of nodes, the **Status** of the instance group updates. When the change you requested is complete, the **Status** is **Running**.

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 `InstanceId`, 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-2AXXXXXXXGAPLF` with the cluster ID.

```
aws emr describe-cluster --cluster-id j-2AXXXXXXXGAPLF
```

Using the AWS CLI, you can also terminate an instance in the core instance group with the `--modify-instance-groups` subcommand.

Warning

Specifying `EC2InstanceIdsToTerminate` must be done with caution. Instances are terminated immediately, regardless of the status of applications running on them, and the instance is not automatically replaced. This is true regardless of the cluster's **Scale down behavior** configuration. Terminating an instance in this way risks data loss and unpredictable cluster behavior.

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-f9XXXXXf2` with the instance ID.

```
aws emr modify-instance-groups --instance-groups  
  InstanceGroupId=ig-6RXXXXXX07SA, EC2InstanceIdsToTerminate=i-f9XXXXXf2
```

For more information about using Amazon EMR commands in the AWS CLI, see <https://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 from 1–48 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 five 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=m4.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 five task instance groups in a single command.

```
aws emr add-instance-groups --cluster-id j-JXBXXXXXX37R --instance-groups  
  InstanceCount=6,InstanceGroupType=task,InstanceType=m4.large  
  InstanceCount=10,InstanceGroupType=task,InstanceType=m4.large
```

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

Interrupting a Resize

Using Amazon EMR version 4.1.0 or later, 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**.



To interrupt a resize using the AWS CLI

You can use the AWS CLI to stop a resize by using the `modify-instance-groups` subcommand. Assume that 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-myInstanceId,InstanceCount=10
```

The second request to stop the first request:

```
aws emr modify-instance-groups --instance-groups  
  InstanceGroupId=ig-myInstanceId,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 an 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": "m4.large",
                "EbsConfiguration": {
                    "VolumeSpecification": {
                        "VolumeType": "Standard"
                    },
                    "EbsBlockDevices": [
                        {
                            "VolumeSize": 100
                        }
                    ]
                }
            }
        ]
    }
}
```

```
        "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": "m4.large",
        "Id": "ig-3SUXXXXXXQ9ZM",
        "Market": "ON_DEMAND",
        "RunningInstanceCount": 1
    }
}
...
}
```

To view information about 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
```

Cluster Scale-Down

With Amazon EMR release version 5.1.0 and later, there are two options for scale-down behavior: terminate at the instance-hour boundary for Amazon EC2 billing, or terminate at task completion. Starting with Amazon EMR release version 5.10.0, the setting for termination at instance-hour boundary is deprecated because of the introduction of per-second billing in Amazon EC2. We do not recommend specifying termination at the instance-hour boundary in versions where the option is available.

Warning

If you use the AWS CLI to issue a `modify-instance-groups` with `EC2InstanceIdsToTerminate`, these instances are terminated immediately, without consideration for these settings, and regardless of the status of applications running on them. Terminating an instance in this way risks data loss and unpredictable cluster behavior.

When terminate at task completion is specified, Amazon EMR blacklists and drains tasks from nodes before terminating the Amazon EC2 instances. With either behavior specified, Amazon EMR does not terminate Amazon EC2 instances in core instance groups if it could lead to HDFS corruption.

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.

This time is set using a property in the `yarn-site` configuration classification. Using Amazon EMR release version 5.12.0 and later, specify the `yarn.resourcemanager.nodemanager-graceful-decommission-timeout-secs` property. Using earlier Amazon EMR release versions, specify the `yarn.resourcemanager.decommissioning.timeout` property.

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). You can set this timeout to be an arbitrarily high value to force graceful shrink to wait longer. For more information, see [Graceful Decommission of YARN Nodes](#) in Apache Hadoop documentation.

Task Node Groups

Amazon EMR intelligently selects instances that are not running tasks related to any step or application, and removes them from a cluster first. If all instances in the cluster are being used, Amazon EMR waits 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 dynamically uses 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.

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**.

The screenshot shows the 'Create Cluster - Advanced Options' page. On the left, there's a sidebar with steps: Step 1: Software and Steps, Step 2: Hardware, Step 3: General Cluster Settings (which is selected and highlighted in orange), and Step 4: Security. The main area is titled 'General Options'. It includes fields for 'Cluster name' (set to 'My cluster'), checkboxes for 'Logging' (S3 folder: s3://aws-logs-176430881729-us-east-1/elasticmapredu/), 'Debugging', and 'Termination protection', and a dropdown menu for 'Scale down behavior' (set to 'Terminate at instance hour').

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`.

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 \(p. 270\)](#).

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 \(p. 270\)](#)
- [Submit Hadoop Jobs Interactively \(p. 272\)](#)
- [Add More than 256 Steps to a Cluster \(p. 274\)](#)

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=m4.large \
  InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large \
  --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=m4.large
  InstanceGroupType=CORE,InstanceCount=2,InstanceType=m4.large --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 <https://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 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 --cluster-id j-2QUAJ7T3OTEI8 --step-ids 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 \(p. 211\)](#).

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 <https://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-2A6HXXXXXXL7J --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 <https://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 \(p. 270\)](#).

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 \(p. 239\)](#). For more information about interactively submitting Hadoop jobs, see [Submit Hadoop Jobs Interactively \(p. 272\)](#).

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

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? \(p. 275\)](#)
- [Viewing and Restarting Amazon EMR and Application Processes \(Daemons\) \(p. 276\)](#)
- [Troubleshoot a Failed Cluster \(p. 278\)](#)
- [Troubleshoot a Slow Cluster \(p. 281\)](#)
- [Common Errors in Amazon EMR \(p. 287\)](#)

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 \(p. 126\)](#) and [Step 5: Test the Cluster Step by Step \(p. 281\)](#).

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 \(p. 275\)](#)
- [Tools to View Log Files \(p. 276\)](#)
- [Tools to Monitor Cluster Performance \(p. 276\)](#)

Tools to Display Cluster Details

You can use the AWS Management Console, AWS CLI, or EMR API to retrieve detailed information about an EMR cluster and job execution. For more information about using the AWS Management Console and AWS CLI, see [View Cluster Status and Details \(p. 203\)](#).

Amazon EMR Console Details Pane

In the **Clusters** list on the Amazon EMR console you can see high-level information about the status of each cluster in your account and region. The list displays all clusters that you have launched in the past two months, regardless of whether they are active or terminated. From the **Clusters** list, you can select a cluster **Name** to view cluster details. This information is organized in different categories to make it easy to navigate.

The **Application history** available in the cluster details page can be particularly useful for troubleshooting. It provides status of YARN applications, and for some, such as Spark applications you can drill into different metrics and facets, such as jobs, stages, and executors. For more information, see [View Application History \(p. 210\)](#). This feature is available only in Amazon EMR version 5.8.0 and later.

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 \(p. 126\)](#).

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 \(p. 243\)](#).

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 \(p. 223\)](#).

Viewing and Restarting Amazon EMR and Application Processes (Daemons)

When you troubleshoot a cluster, you may want to list running processes. You may also find it useful to stop or restart processes in some circumstances—for example, after you change a configuration or notice a problem with a particular process after you analyze log files and error messages.

There are two types of processes that run on a cluster: Amazon EMR processes (for example, instance-controller and Log Pusher), and processes associated with the applications installed on the cluster (for example, hadoop-hdfs-namenode, and hadoop-yarn-resourcemanager).

To work with processes directly on a cluster, you connect to the master node. For more information, see [Connect to the Cluster \(p. 238\)](#).

Viewing Running Processes

If you are using Amazon EMR version 4.x or later, application releases are packaged using a system based on Apache Bigtop, so these application processes are configured via .conf scripts under the upstart init system. Amazon EMR processes, on the other hand, are configured using SysV (init.d scripts) which is backwards compatible with upstart.

To view a list of running Amazon EMR processes

- Type the following command (without the \$, which indicates the Linux command prompt):

```
$ ls /etc/init.d/
```

The command returns a list of running Amazon EMR processes similar to the following example:

acpid	cloud-init-local	instance-controller	ntpd
-------	------------------	---------------------	------

To view a list of processes associated with application releases

- Type the following command:

```
$ ls /etc/init/
```

The command returns a list of running application processes similar to the following example:

control-alt-delete.conf	hadoop-yarn-resourcemanager.conf	hive-
metastore.conf		

Restarting Processes

After you determine which processes are running, you can stop and then restart them if necessary. How you start and stop a service depends on whether it's an Amazon EMR service or a service associated with an application.

To restart a process associated with an application release

- Type the following command to stop the process, replacing *processname* with the process name returned by the ls command in the procedure above:

```
$ sudo /etc/init.d/processname stop
```

For example: sudo /etc/init.d/hadoop-hdfs-namenode stop

- Type the following command to restart the process:

```
$ sudo /etc/init.d/processname start
```

For example, sudo /etc/init.d/hadoop-hdfs-namenode start.

To restart an Amazon EMR process

1. Type the following command to stop the process, replacing *processname* with the process name returned by the `ls` command in the procedure above:

```
$ sudo /sbin/stop processname
```

For example, `sudo /sbin/stop instance-controller`.

2. Type the following command to restart the process:

```
$ sudo sbin/start processname
```

For example, `sudo sbin/start instance-controller`.

Note

The `sbin/start`, `stop` and `restart` commands are symlinks to `/sbin/initctl`. For more information about `initctl`, see the `initctl` man page by typing `man initctl` at the command prompt.

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 \(p. 281\)](#) instead.

Topics

- [Step 1: Gather Data About the Issue \(p. 278\)](#)
- [Step 2: Check the Environment \(p. 279\)](#)
- [Step 3: Look at the Last State Change \(p. 280\)](#)
- [Step 4: Examine the Log Files \(p. 280\)](#)
- [Step 5: Test the Cluster Step by Step \(p. 281\)](#)

Step 1: Gather Data About the Issue

The first step in troubleshooting a 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 Status and Details \(p. 203\)](#).

- 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 \(p. 279\)](#)
- [Check Usage Limits \(p. 279\)](#)
- [Check the Release Version \(p. 279\)](#)
- [Check the Amazon VPC Subnet Configuration \(p. 280\)](#)

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 [Configure Networking \(p. 102\)](#). 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 Status and Details \(p. 203\)](#).

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 \(p. 211\)](#).

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.

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 [Configuring a Cluster to Auto-Terminate or Continue \(p. 79\)](#) and [Using Termination Protection \(p. 79\)](#).
2. Submit a step to the cluster. For more information, see [Submit Work to a Cluster \(p. 270\)](#).
3. When the step completes processing, check for errors in the step log files. For more information, see [Step 4: Examine the Log Files \(p. 280\)](#). 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 [Using Termination Protection \(p. 79\)](#).

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 \(p. 278\)](#)

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 \(p. 96\)](#).

The following topics walk you through the process of identifying alternative causes of a slow cluster.

Topics

- [Step 1: Gather Data About the Issue \(p. 282\)](#)
- [Step 2: Check the Environment \(p. 282\)](#)
- [Step 3: Examine the Log Files \(p. 283\)](#)
- [Step 4: Check Cluster and Instance Health \(p. 284\)](#)
- [Step 5: Check for Arrested Groups \(p. 285\)](#)
- [Step 6: Review Configuration Settings \(p. 286\)](#)
- [Step 7: Examine Input Data \(p. 287\)](#)

Step 1: Gather Data About the Issue

The first step in troubleshooting a 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 Status and Details \(p. 203\)](#).

- 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 \(p. 282\)](#)
- [Check Usage Limits \(p. 283\)](#)
- [Check the Amazon VPC Subnet Configuration \(p. 283\)](#)
- [Restart the Cluster \(p. 283\)](#)

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 [Configure Networking \(p. 102\)](#). 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 \(p. 211\)](#).

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 \(p. 262\)](#).
- **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 \(p. 262\)](#).

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 \(p. 223\)](#).

Check Job Status and HDFS Health

Use the **Application history** on the cluster details page to view YARN application details. For certain applications, you can drill into further detail and access logs directly. This is particularly useful for Spark applications. For more information, see [View Application History \(p. 210\)](#).

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 \(p. 243\)](#).

- 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 \(p. 215\)](#).

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 core and task 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 \(p. 266\)](#).

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 [Create Bootstrap Actions to Install Additional Software \(p. 92\)](#). 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

Configuration Setting	Description
	TaskTracker uses when launching a JVM for a task to execute within. A common parameter is "-Xmx" for setting max memory size.
mapred.map.child.java.opts	The Java options TaskTracker uses when launching a JVM for a map task to execute within. A common parameter is "-Xmx" for setting max memory heap size.
mapred.map.tasks.speculative.execution	Determines whether map task attempts of the same task may be launched in parallel.
mapred.reduce.tasks.speculative.execution	Determines whether reduce task attempts of the same task may be launched in parallel.
mapred.map.max.attempts	The maximum number of times a map task can be attempted. If all fail, then the map task is marked as failed.
mapred.reduce.child.java.opts	The Java options TaskTracker uses when launching a JVM for a reduce task to execute within. A common parameter is "-Xmx" for setting max memory heap size.
mapred.reduce.max.attempts	The maximum number of times a reduce task can be attempted. If all fail, then the map task is marked as failed.
mapred.reduce.slowstart.completed.maps	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.
mapred.reuse.jvm.num.tasks	A task runs within a single JVM. Specifies how many tasks may reuse the same JVM.
mapred.tasktracker.map.tasks.maximum	The max amount of tasks that can execute in parallel per task node during mapping.
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 \(p. 288\)](#)
- [Permissions Errors \(p. 290\)](#)
- [Resource Errors \(p. 290\)](#)
- [Streaming Cluster Errors \(p. 295\)](#)
- [Custom JAR Cluster Errors \(p. 296\)](#)
- [Hive Cluster Errors \(p. 297\)](#)
- [VPC Errors \(p. 298\)](#)
- [AWS GovCloud \(US-West\) Errors \(p. 300\)](#)
- [Other Issues \(p. 301\)](#)

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? \(p. 288\)](#)
- [Are you trying to recursively traverse input directories? \(p. 288\)](#)
- [Does your output directory already exist? \(p. 288\)](#)
- [Are you trying to specify a resource using an HTTP URL? \(p. 289\)](#)
- [Are you referencing an Amazon S3 bucket using an invalid name format? \(p. 289\)](#)
- [Are you experiencing trouble loading data to or from Amazon S3? \(p. 289\)](#)

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.

- Launch your cluster using the most recent release version of Amazon EMR.
- 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? \(p. 297\)](#).

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? \(p. 290\)](#)
- [If you are using IAM, do you have the proper Amazon EC2 policies set? \(p. 290\)](#)

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 \(p. 239\)](#).

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 \(p. 137\)](#).

Resource Errors

The following errors are commonly caused by constrained resources on the cluster.

Cluster Terminates With NO_SLAVE_LEFT and Core Nodes FAILED_BY_MASTER

Usually, this happens because termination protection is disabled, and all core nodes exceed disk storage capacity as specified by a maximum utilization threshold in the `yarn-site` configuration classification, which corresponds to the `yarn-site.xml` file. This value is 90% by default. When disk utilization for a core node exceeds the utilization threshold, the YARN NodeManager health service reports the node as `UNHEALTHY`. While it's in this state, Amazon EMR blacklists the node and does not allocate YARN containers to it. If the node remains unhealthy for 45 minutes, Amazon EMR marks the associated Amazon EC2 instance for termination as `FAILED_BY_MASTER`. When all Amazon EC2 instances associated with core nodes are marked for termination, the cluster terminates with the status `NO_SLAVE_LEFT` because there are no resources to execute jobs.

Exceeding disk utilization on one core node might lead to a chain reaction. If a single node exceeds the disk utilization threshold because of HDFS, other nodes are likely to be near the threshold as well. The first node exceeds the disk utilization threshold, so Amazon EMR blacklists it. This increases the burden of disk utilization for remaining nodes because they begin to replicate HDFS data among themselves that they lost on the blacklisted node. Each node subsequently goes `UNHEALTHY` in the same way, and the cluster eventually terminates.

Best Practices and Recommendations

Configure Cluster Hardware with Adequate Storage

When you create a cluster, make sure that there are enough core nodes and that each has an adequate instance store and EBS storage volumes for HDFS. For more information, see [Calculating the Required HDFS Capacity of a Cluster \(p. 126\)](#). You can also add core instances to existing instance groups manually or by using auto-scaling. The new instances have the same storage configuration as other instances in the instance group. For more information, see [Scaling Cluster Resources \(p. 253\)](#).

Enable Termination Protection

Enable termination protection. This way, if a core node is blacklisted, you can connect to the associated Amazon EC2 instance using SSH to troubleshoot and recover data. If you enable termination protection, be aware that Amazon EMR does not replace the Amazon EC2 instance with a new instance. For more information, see [Using Termination Protection \(p. 79\)](#).

Create an Alarm for the MRUnhealthyNodes CloudWatch Metric

This metric reports the number of nodes reporting an `UNHEALTHY` status. It's equivalent to the YARN metric `mapred.resourcemanager.NoOfUnhealthyNodes`. You can set up a notification for this alarm to warn you of unhealthy nodes before the 45-minute timeout is reached. For more information, see [Monitor Metrics with CloudWatch \(p. 223\)](#).

Tweak Settings Using `yarn-site`

The settings below can be adjusted according to your application requirements. For example, you may want to increase the disk utilization threshold where a node reports `UNHEALTHY` by increasing the value of `yarn.nodemanager.disk-health-checker.max-disk-utilization-per-disk-percentage`.

You can set these values when you create a cluster using the `yarn-site` configuration classification. For more information see [Configuring Applications](#) in the *Amazon EMR Release Guide*. You can also connect to the Amazon EC2 instances associated with core nodes using SSH, and then add the values in `/etc/hadoop/conf.empty/yarn-site.xml` using a text editor. After making the change, you must restart `hadoop-yarn-nodemanager` as shown below.

Important

When you restart the NodeManager service, active YARN containers are killed unless `yarn.nodemanager.recovery.enabled` is set to `true` using the `yarn-site` configuration

classification when you create the cluster. You must also specify the directory in which to store container state using the `yarn.nodemanager.recovery.dir` property.

```
sudo /sbin/stop hadoop-yarn-nodemanager
sudo /sbin/start hadoop-yarn-nodemanager
```

For more information about current `yarn-site` properties and default values, see [YARN default settings](#) in Apache Hadoop documentation.

Property	Default Value	Description
<code>yarn.nodemanager.disk-health-checker.interval-ms</code>	120000	The frequency (in seconds) that the disk health checker runs.
<code>yarn.nodemanager.disk-health-checker.min-healthy-disks</code>	0.25	The minimum fraction of the number of disks that must be healthy for NodeManager to launch new containers. This correspond to both <code>yarn.nodemanager.local-dirs</code> (by default, <code>/mnt/yarn</code> in Amazon EMR) and <code>yarn.nodemanager.log-dirs</code> (by default <code>/var/log/hadoop-yarn/containers</code> , which is symlinked to <code>mnt/var/log/hadoop-yarn/containers</code> in Amazon EMR).
<code>yarn.nodemanager.disk-health-checker.max-disk-utilization-per-disk-percentage</code>	90.0	The maximum percentage of disk space utilization allowed after which a disk is marked as bad. Values can range from 0.0 to 100.0. If the value is greater than or equal to 100, the NodeManager checks for a full disk. This applies to <code>yarn.nodemanager.local-dirs</code> and <code>yarn.nodemanager.log-dirs</code> .
<code>yarn.nodemanager.disk-health-checker.min-free-space-per-disk-mb</code>	0	The minimum space that must be available on a disk for it to be used. This applies to <code>yarn.nodemanager.local-dirs</code> and <code>yarn.nodemanager.log-dirs</code> .

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 nodes X 850 GB per node)/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 to 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 [Create Bootstrap Actions to Install Additional Software \(p. 92\)](#).
- 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 \(p. 96\)](#).
- 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. 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 \(p. 262\)](#) and [Monitor Metrics with CloudWatch \(p. 223\)](#).

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 \(p. 211\)](#).

Are your nodes being blacklisted?

The NodeManager daemon is responsible for launching and managing containers on core and task nodes. The containers are allocated to the NodeManager daemon by the ResourceManager daemon that runs on the master node. The ResourceManager monitors the NodeManager node through a heartbeat.

There are a couple of situations in which the ResourceManager daemon blacklists a NodeManager, removing it from the pool of nodes available to process tasks:

- If the NodeManager has not sent a heartbeat to the ResourceManager daemon in the past 10 minutes (60000 milliseconds). This time period can be configured using the `yarn.nm.liveness-monitor.expiry-interval-ms` configuration setting. For more information about changing Yarn configuration settings, see [Configuring Applications in the Amazon EMR Release Guide](#).
- NodeManager checks the health of the disks determined by `yarn.nodemanager.local-dirs` and `yarn.nodemanager.log-dirs`. The checks include permissions and free disk space (< 90%). If a disk fails the check, the NodeManager stops using that particular disk but still reports the node status as healthy. If a number of disks fail the check, the node is reported as unhealthy to the ResourceManager and new containers are not assigned to the node.

The application master can also blacklist a NodeManager node if it has more than three failed tasks. You can change this to a higher value using the `mapreduce.job.maxtaskfailures.per.tracker` configuration parameter. Other configuration settings you might change control how many times to attempt a task before marking it as failed: `mapreduce.map.max.attempts` for map tasks and `mapreduce.reduce.maxattempts` for reduce tasks. For more information about changing configuration settings, see [Configuring Applications in the Amazon EMR Release Guide](#).

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? \(p. 295\)](#)
- [Is your script timing out? \(p. 295\)](#)
- [Are you passing in invalid streaming arguments? \(p. 296\)](#)
- [Did your script exit with an error? \(p. 296\)](#)

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 \(p. 211\)](#).

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 \(p. 211\)](#).

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 \(p. 211\)](#).

Custom JAR Cluster Errors

The following errors are common to custom JAR clusters.

Topics

- [Is your JAR throwing an exception before creating a job? \(p. 296\)](#)
- [Is your JAR throwing an error inside a map task? \(p. 296\)](#)

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 \(p. 211\)](#).

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 \(p. 211\)](#).

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? \(p. 297\)](#)
- [Did you encounter a syntax error in the Hive script? \(p. 297\)](#)
- [Did a job fail when running interactively? \(p. 297\)](#)
- [Are you having trouble loading data to or from Amazon S3 into Hive? \(p. 297\)](#)

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 \(p. 211\)](#).

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 \(p. 211\)](#).

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? \(p. 289\)](#). 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.
- If you use Amazon EMR release versions earlier than 5.0, you can use the following command in HiveQL to pre-cache the results of an Amazon S3 list operation locally on the cluster:

```
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 \(p. 298\)](#)
- [Missing DHCP Options Set \(p. 298\)](#)
- [Permissions Errors \(p. 299\)](#)
- [Errors That Result in START_FAILED \(p. 299\)](#)
- [Cluster Terminated with errors and NameNode Fails to Start \(p. 299\)](#)

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-West) 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 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-West) Errors

The AWS GovCloud (US-West) 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-West) 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 Permissions to AWS Services \(p. 187\)](#).
- 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 \(p. 298\)](#).

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 \(p. 287\)](#).

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

Topics

- [End-to-End Amazon EMR Java Source Code Sample \(p. 302\)](#)
- [Common Concepts for API Calls \(p. 304\)](#)
- [Use SDKs to Call Amazon EMR APIs \(p. 306\)](#)

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 \(p. 306\)](#).

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 \(p. 306\)](#).

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.AWS Credentials;
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) {

        AWS Credentials 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-39BLQZRJB2E5E, s-1L6A4ZU2SAURC]
```

Common Concepts for API Calls

Topics

- [Endpoints for Amazon EMR \(p. 305\)](#)
- [Specifying Cluster Parameters in Amazon EMR \(p. 305\)](#)

- [Availability Zones in Amazon EMR \(p. 305\)](#)
- [How to Use Additional Files and Libraries in Amazon EMR Clusters \(p. 306\)](#)

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.

Use SDKs to Call Amazon EMR APIs

Topics

- [Using the AWS SDK for Java to Create an Amazon EMR Cluster \(p. 306\)](#)
- [Using the Java SDK to Sign an API Request \(p. 307\)](#)

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 enabledebugging = 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(enabledebugging)
    .withApplications(myApp)
    .withLogUri("s3://myawsbucket/")
    .withServiceRole("service_role")
    .withJobFlowRole("jobflow_role")
    .withInstances(new JobFlowInstancesConfig()
        .withEc2KeyName("keypair")
        .withInstanceCount(5)
        .withKeepJobFlowAliveWhenNoSteps(true)
        .withMasterInstanceType("m4.large")
        .withSlaveInstanceType("m4.large")));
    .withSlaveInstanceType("m4.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::AccountID:role/EMR_DefaultRole"
"RoleName": "EMR_EC2_DefaultRole",
"Arn": "arn:aws:iam::AccountID: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*.

AWS Glossary

For the latest AWS terminology, see the [AWS Glossary](#) in the *AWS General Reference*.