Amazon EMR

Management Guide

Amazon EMR: Management Guide

Copyright © 2022 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 ... 5
  - Cost savings ... 5
  - AWS integration ... 6
  - Deployment ... 6
  - Scalability and flexibility ... 6
  - Reliability ... 7
  - Security ... 7
  - Monitoring ... 8
  - Management interfaces ... 8
- Architecture ... 9
  - Storage ... 9
  - Cluster resource management ... 10
  - Data processing frameworks ... 10
  - Applications and programs ... 11

Setting up Amazon EMR ... 12
- Sign up for AWS ... 12
- Create an Amazon EC2 key pair for SSH ... 12
- Next steps ... 12

Getting started tutorial ... 14
- Overview ... 14
- Step 1: Plan and configure ... 15
  - Prepare storage for Amazon EMR ... 15
  - Prepare an application with input data for Amazon EMR ... 15
  - Launch an Amazon EMR cluster ... 17
- Step 2: Manage ... 18
  - Submit work to Amazon EMR ... 18
  - View results ... 21
- Step 3: Clean up ... 23
  - Terminate your cluster ... 23
  - Delete S3 resources ... 24
- Next steps ... 25
  - Explore big data applications for Amazon EMR ... 25
  - Plan cluster hardware, networking, and security ... 25
  - Manage clusters ... 25
  - Use a different interface ... 25
  - Browse the EMR technical blog ... 25

Amazon EMR Studio ... 26
- Key features of EMR Studio ... 26
- How Amazon EMR Studio works ... 26
  - Authentication and user login ... 27
  - Access control ... 29
  - Workspaces ... 30
  - Notebook storage ... 30
- Considerations and limitations ... 30
  - Known issues ... 31
  - Feature limitations ... 32
  - Service limits ... 32
  - VPC and subnet best practices ... 33

Cluster requirements ... 33
Configure EMR Studio ... 34
Administrator permissions to create an EMR Studio ... 35
Set up an Amazon EMR Studio ... 38
Manage a Studio ... 69
Control EMR Studio network traffic ... 73
Create cluster templates ... 75
Access and permissions for Git-based repositories ... 79
Optimize Spark jobs ... 81
Use an Amazon EMR Studio ... 82
Workspace basics ... 82
Configure Workspace collaboration ... 86
Run Workspace notebooks programmatically ... 88
Browse data with SQL Explorer ... 88
Attach a cluster to a Workspace ... 89
Link Git repositories ... 92
Debug applications and jobs ... 94
Install kernels and libraries ... 97
Enhance kernels with magic commands ... 97
Use multi-language notebooks with Spark kernels ... 102
EMR Notebooks ... 104
Considerations ... 104
Cluster requirements ... 104
Differences in capabilities by cluster release version ... 105
Limits for concurrently attached EMR Notebooks ... 106
Jupyter Notebook and Python versions ... 106
... 106
Creating a Notebook ... 106
Working with EMR Notebooks ... 108
Understanding Notebook status ... 108
Working with the Notebook editor ... 109
Changing clusters ... 110
Deleting Notebooks and Notebook files ... 110
Sharing Notebook files ... 111
Executing EMR Notebooks programmatically ... 111
CLI command samples ... 112
Boto3 SDK sample script ... 115
Ruby sample script ... 117
User impersonation for Spark ... 118
Setting up Spark user impersonation ... 119
Using the Spark job monitoring widget ... 119
Security ... 120
Installing and using kernels and libraries ... 121
Installing kernels and Python libraries on a cluster master node ... 121
Using Notebook-scoped libraries ... 123
Working with Notebook-scoped libraries ... 123
Associating Git-based repositories with EMR Notebooks ... 124
Prerequisites and considerations ... 125
Add a Git-based repository to Amazon EMR ... 126
Update or delete a Git-based repository ... 128
Link or unlink a Git-based repository ... 128
Create a new Notebook with an associated Git repository ... 129
Use Git repositories in a Notebook ... 130
Plan and configure clusters ... 131
Launch a cluster quickly ... 131
Configure cluster location and data storage ... 131
Choose an AWS Region ... 132

Work with storage and file systems ... 133
Prepare input data ... 135
Configure an output location ... 144
Plan and configure master nodes ... 148
Supported applications and features ... 149
Launch an Amazon EMR Cluster with multiple master nodes ... 154
Amazon EMR integration with EC2 placement groups ... 156
Considerations and best practices ... 160
EMR clusters on AWS Outposts ... 160
Prerequisites ... 161
Limitations ... 161
Network connectivity considerations ... 161
Creating an Amazon EMR cluster on AWS Outposts ... 162
EMR clusters on AWS Local Zones ... 162
Supported instance types ... 163
Creating an Amazon EMR cluster on Local Zones ... 163
Configure Docker ... 164
Docker registries ... 164
Configuring Docker registries ... 165
Configuring YARN to access Amazon ECR on EMR 6.0.0 and earlier ... 165
Control cluster termination ... 167
Configuring a cluster to continue or terminate after step execution ... 167
Using an auto-termination policy ... 169
Using termination protection ... 172
Working with AMIs ... 176
Using the default AMI ... 177
Using a custom AMI ... 181
Changing the Amazon Linux release for a launching cluster ... 188
Specifying the Amazon EBS root device volume size ... 189
Configure cluster software ... 190
Create bootstrap actions to install additional software ... 191
Configure cluster hardware and networking ... 195
Understand node types ... 195
Configure Amazon EC2 instances ... 197
Configure networking ... 306
Configure instance fleets or instance groups ... 315
Configure cluster logging and debugging ... 342
Default log files ... 342
Archive log files to Amazon S3 ... 343
Log locations ... 346
Enable the debugging tool ... 347
Debugging option information ... 348
Tag clusters ... 348
Tag restrictions ... 349
Tag resources for billing ... 349
Add tags to a new cluster ... 350
Adding tags to an existing cluster ... 350
View tags on a cluster ... 351
Remove tags from a cluster ... 351
Drivers and third-party application integration ... 352
Use business intelligence tools with Amazon EMR ... 352
Security ... 353
Security configurations ... 353
Data protection ... 353
AWS Identity and Access Management with Amazon EMR ... 353
Kerberos ... 354
Lake Formation ... 354

Secure Socket Shell (SSH) ... 354
Amazon EC2 security groups ... 354
Default Amazon Linux AMI updates ... 354
Use security configurations to set up cluster security ... 355
Create a security configuration ... 355
Specify a security configuration for a cluster ... 373
Data protection ... 373
Encrypt data at rest and in transit ... 374
IAM with Amazon EMR ... 382
Audience ... 383
Authenticating with identities ... 383
Managing access using policies ... 385
How Amazon EMR works with IAM ... 386
Configure runtime roles for Amazon EMR steps ... 388
Configure service roles for Amazon EMR ... 394
Identity-based policy examples ... 426
Authenticate to cluster nodes ... 447
Use an Amazon EC2 key pair for SSH credentials ... 447
Use Kerberos authentication ... 448
Integrate Amazon EMR with AWS Lake Formation ... 473
Overview ... 473
Applications, features, and limitations ... 479
Before you begin ... 480
Launch an Amazon EMR cluster with Lake Formation ... 489
Integrate Amazon EMR with Apache Ranger ... 495
Ranger overview ... 495
Application support and limitations ... 498
Set up Amazon EMR for Apache Ranger ... 499
Apache Ranger plugins ... 511
Apache Ranger troubleshooting ... 530
Control network traffic with security groups ... 532
Working with Amazon EMR-managed security groups ... 533
Working with additional security groups ... 539
Specifying security groups ... 539
Security groups for EMR Notebooks ... 541
Using block public access ... 542
Compliance validation ... 545
Resilience ... 545
Infrastructure security ... 546
Connect to Amazon EMR using an interface VPC endpoint ... 546
Manage clusters ... 550
View and monitor a cluster ... 550
View cluster status and details ... 550
Enhanced step debugging ... 556
View application history ... 557
View log files ... 564
View cluster instances in Amazon EC2 ... 568
CloudWatch events and metrics ... 568
View cluster application metrics with Ganglia ... 589
Logging Amazon EMR API calls in AWS CloudTrail ... 589
Connect to the cluster ... 591
Before you connect ... 592
Connect to the master node using SSH ... 593
Terminate a cluster ... 607
Terminate a cluster using the console ... 608
Terminate a cluster using the AWS CLI ... 608
Terminate a cluster using the API ... 609

Scaling cluster resources ... 610
Using EMR managed scaling in Amazon EMR ... 610
Using automatic scaling with a custom policy for instance groups ... 625
Manually resizing a running cluster ... 634
Cluster scale-down ... 640
Cloning a cluster using the console ... 641
Submit work to a cluster ... 642
Adding steps to a cluster using the console ... 642
Adding steps to a cluster using the AWS CLI ... 644
Considerations for running multiple steps in parallel ... 645
Viewing steps ... 646
Canceling steps ... 646
Automate recurring clusters with AWS Data Pipeline ... 647
Troubleshoot a cluster ... 648
What tools are available for troubleshooting? ... 648
Tools to display cluster details ... 648
Tools to run scripts and configure processes ... 649
Tools to view log files ... 649
Tools to monitor cluster performance ... 649
Viewing and restarting Amazon EMR and application processes (daemons) ... 650
Viewing running processes ... 650
Stopping and restarting processes ... 651
Troubleshoot a failed cluster ... 653
Step 1: Gather data about the issue ... 654
Step 2: Check the environment ... 654
Step 3: Look at the last state change ... 655
Step 4: Examine the log files ... 655
Step 5: Test the cluster step by step ... 656
Troubleshoot a slow cluster ... 657
Step 1: Gather data about the issue ... 657
Step 2: Check the environment ... 658
Step 3: Examine the log files ... 659
Step 4: Check cluster and instance health ... 660
Step 5: Check for suspended groups ... 661
Step 6: Review configuration settings ... 661
Step 7: Examine input data ... 663
Common errors in Amazon EMR ... 663
Input and output errors ... 663
Permissions errors ... 665
Resource errors ... 666
Streaming cluster errors ... 673
Custom JAR cluster errors ... 674
Hive cluster errors ... 674
VPC errors ... 675
AWS GovCloud (US-West) errors ... 678
Other issues ... 678
Troubleshoot a Lake Formation cluster ... 679
Data lake access not allowed ... 679
Session expiration ... 679
No permissions for user on requested table ... 679
Inserting into, creating and altering tables: Unsupported in beta ... 679
Write applications that launch and manage clusters ... 681
End-to-end Amazon EMR Java source code sample ... 681
Common concepts for API calls ... 683
Endpoints for Amazon EMR ... 684
Specifying cluster parameters in Amazon EMR ... 684
Availability Zones in Amazon EMR ... 684

How to use additional files and libraries in Amazon EMR clusters 685
Use SDKs to call Amazon EMR APIs 685
Using the AWS SDK for Java to create an Amazon EMR cluster 685
Manage Amazon EMR Service Quotas 687
What are Amazon EMR Service Quotas 687
How to manage Amazon EMR Service Quotas 688
When to set up EMR events in CloudWatch 688
AWS glossary 691

What is Amazon EMR?

Amazon EMR (previously called Amazon Elastic MapReduce) 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. Using these frameworks and related open-source projects, you can process data for analytics purposes and business intelligence workloads. Amazon EMR also lets you 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 Amazon EMR highlights, product details, and pricing information.
- Tutorial: Getting started with Amazon EMR (p. 14) – This tutorial gets you started using Amazon EMR quickly.

In This Section

- Overview of Amazon EMR (p. 1)
- Benefits of using Amazon EMR (p. 5)
- Overview of Amazon EMR architecture (p. 9)

Overview of Amazon EMR

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

In This Topic

- Understanding clusters and nodes (p. 1)
- Submitting work to a cluster (p. 2)
- Processing data (p. 2)
- Understanding the cluster lifecycle (p. 3)

Understanding clusters and nodes

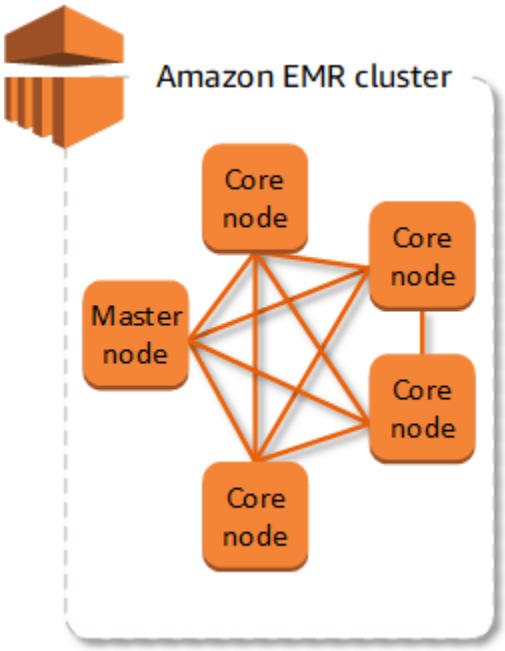
The central component of Amazon EMR is the *cluster*. A cluster is a collection of Amazon Elastic Compute Cloud (Amazon EC2) instances. Each instance in the cluster is called a *node*. Each node has a role within the cluster, referred to as the *node type*. Amazon EMR also installs different software 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. 642).
- 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. 591).

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:

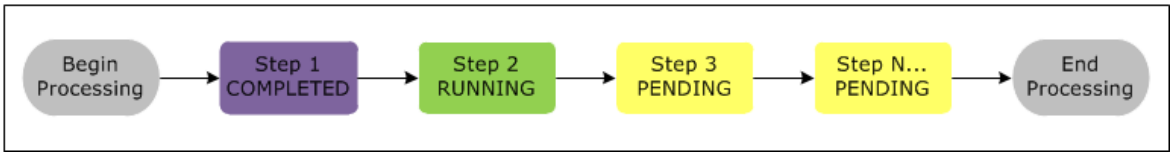
1. Submit an input dataset for processing.
2. Process the output of the first step by using a Pig program.
3. Process a second input dataset by using a Hive program.
4. Write an output dataset.

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:

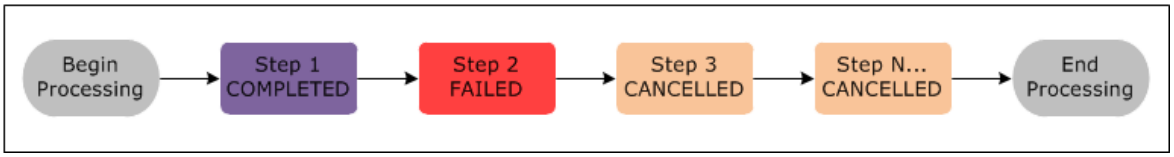
1. A request is submitted to begin processing steps.
2. The state of all steps is set to **PENDING**.
3. When the first step in the sequence starts, its state changes to **RUNNING**. The other steps remain in the **PENDING** state.
4. After the first step completes, its state changes to **COMPLETED**.
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 **FAILED**. 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 if a preceeding step fails. 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.



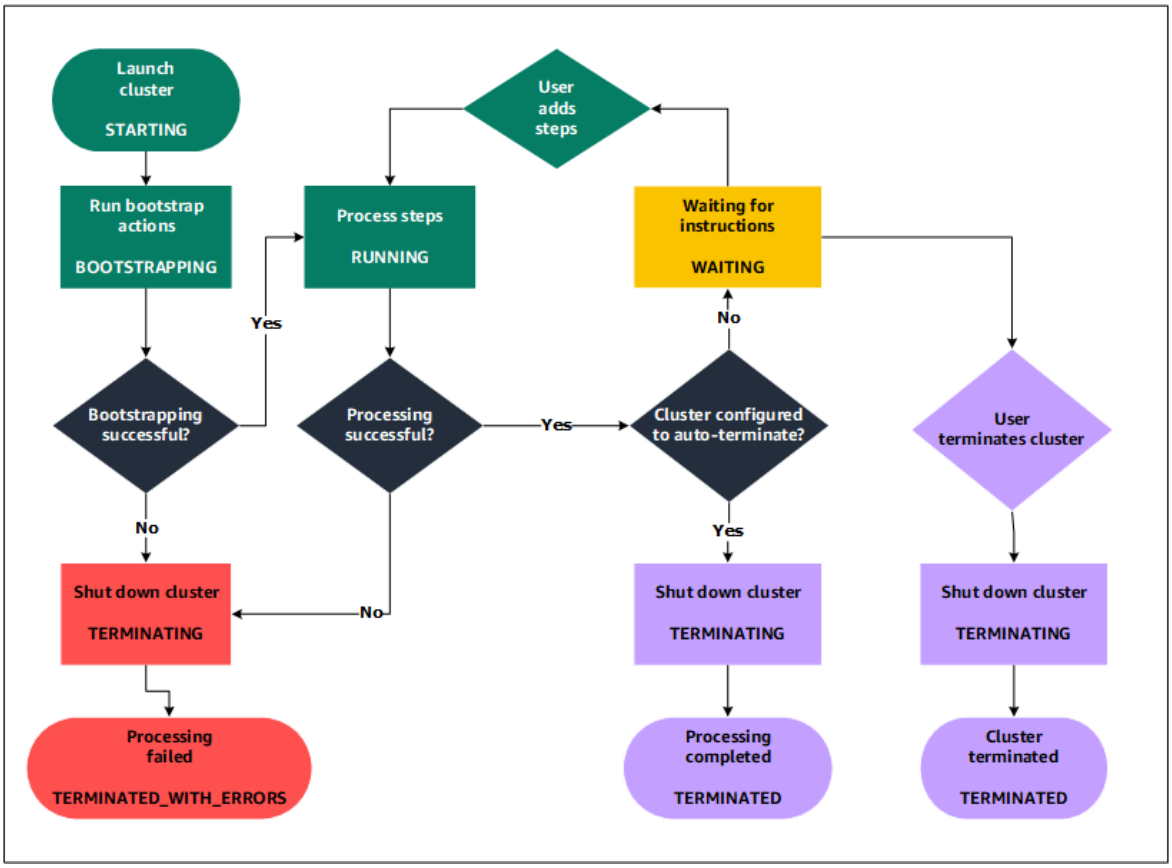
Understanding the cluster lifecycle

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. 195). 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. 181). 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. 191). 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 Submit work to a cluster (p. 642).
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 `TERMINATING` state and then into the `TERMINATED` state. If the cluster is configured to wait, you must manually shut it down when you no longer need it. After you manually shut down the cluster, it goes into the `TERMINATING` state and then into the `TERMINATED` 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 `TERMINATED_WITH_ERRORS`. 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. 172).

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



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. 5)
- AWS integration (p. 6)
- Deployment (p. 6)
- Scalability and flexibility (p. 6)
- Reliability (p. 7)
- Security (p. 7)
- Monitoring (p. 8)
- Management interfaces (p. 8)

Cost savings

Amazon EMR pricing depends on the instance type and number of Amazon 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.

Note
When you set up an Amazon EMR cluster in a private subnet, we recommend that you also set up VPC endpoints for Amazon S3. If your EMR cluster is in a private subnet without VPC endpoints for Amazon S3, you will incur additional NAT gateway charges that are associated with S3 traffic because the traffic between your EMR cluster and S3 will not stay within your VPC.

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
- AWS Lake Formation to discover, catalog, and secure data in an Amazon S3 data lake

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. 195).

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. 190).

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. 634).

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 at 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. 339).

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. 133).

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 if 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. 172).

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 How Amazon EMR works with IAM (p. 386).

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 service roles for Amazon EMR permissions to AWS services and resources (p. 394).

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. 532).

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 root key for client-side encryption using either the AWS Key Management Service (AWS KMS) or your own key management system.

For more information, see Specifying Amazon S3 encryption using EMRFS properties.

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. 306).

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. 589).

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. 447).

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. 342).

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. 577).

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. 9)
- Cluster resource management (p. 10)
- Data processing frameworks (p. 10)
- Applications and programs (p. 11)

Storage

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

Hadoop Distributed File System (HDFS)

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

For more information, see Instance storage (p. 303) in this guide or go to HDFS User 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 do not 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 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.

Setting up Amazon EMR

Complete the tasks in this section before you launch an Amazon EMR cluster for the first time:

1. Sign up for AWS (p. 12)
2. Create an Amazon EC2 key pair for SSH (p. 12)

Sign up for AWS

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

To sign up for an AWS account

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

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

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

Create an Amazon EC2 key pair for SSH

> **Note**
> With Amazon EMR release versions 5.10.0 or later, you can configure Kerberos to authenticate users and SSH connections to a cluster. For more information, see Use Kerberos authentication (p. 448).

To authenticate and connect to the nodes in a cluster over a secure channel using the Secure Shell (SSH) protocol, create an Amazon Elastic Compute Cloud (Amazon EC2) key pair before you launch the cluster. You can also create a cluster without a key pair. This is usually done with transient clusters that start, run steps, and then terminate automatically.

If...	Then...
You already have an Amazon EC2 key pair that you want to use, or you don't need to authenticate to your cluster.	Skip this step.
You need to create a key pair.	See Creating your key pair using Amazon EC2.

Next steps

- For guidance on creating a sample cluster, see Tutorial: Getting started with Amazon EMR (p. 14).

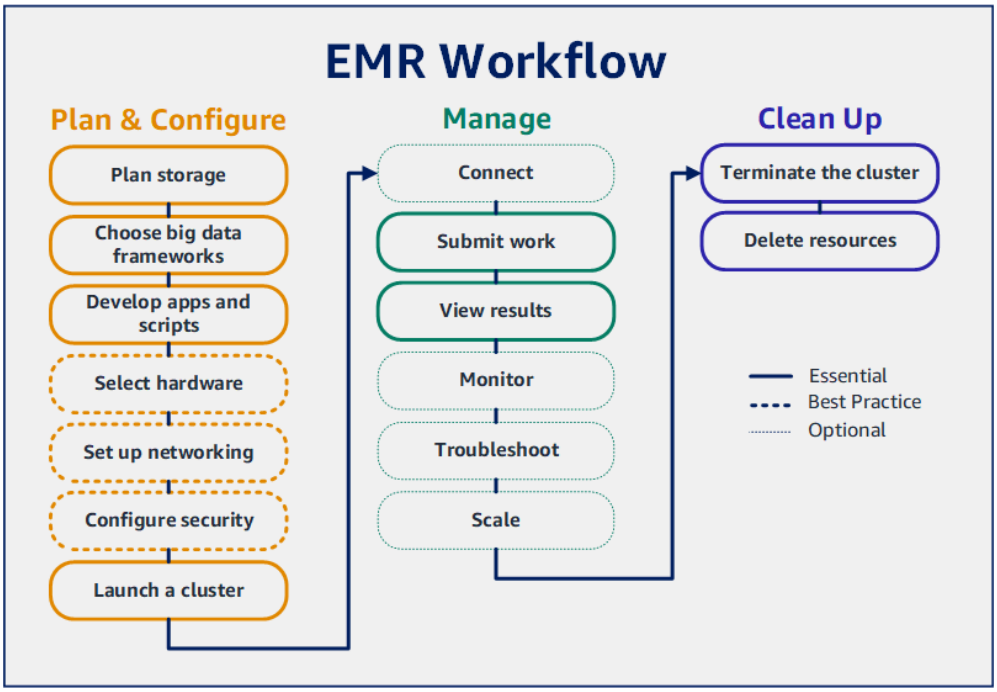
- For more information on how to configure a custom cluster and control access to it, see Plan and configure clusters (p. 131) and Security in Amazon EMR (p. 353).

Tutorial: Getting started with Amazon EMR

Overview

With Amazon EMR you can set up a cluster to process and analyze data with big data frameworks in just a few minutes. This tutorial shows you how to launch a sample cluster using Spark, and how to run a simple PySpark script stored in an Amazon S3 bucket. It covers essential Amazon EMR tasks in three main workflow categories: Plan and Configure, Manage, and Clean Up.

You'll find links to more detailed topics as you work through the tutorial, and ideas for additional steps in the Next steps (p. 25) section. If you have questions or get stuck, contact the Amazon EMR team on our Discussion forum.



Prerequisites

- Before you launch an Amazon EMR cluster, make sure you complete the tasks in Setting up Amazon EMR (p. 12).

Cost

- The sample cluster that you create runs in a live environment. The cluster accrues minimal charges. To avoid additional charges, make sure you complete the cleanup tasks in the last step of this tutorial. Charges accrue at the per-second rate according to Amazon EMR pricing. Charges also vary by Region. For more information, see Amazon EMR pricing.

- Minimal charges might accrue for small files that you store in Amazon S3. Some or all of the 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.

Step 1: Plan and configure an Amazon EMR cluster

Prepare storage for Amazon EMR

When you use Amazon EMR, you can choose from a variety of file systems to store input data, output data, and log files. In this tutorial, you use EMRFS to store data in an S3 bucket. EMRFS is an implementation of the Hadoop file system that lets you read and write regular files to Amazon S3. For more information, see Work with storage and file systems (p. 133).

To create a bucket for this tutorial, follow the instructions in How do I create an S3 bucket? in the *Amazon Simple Storage Service Console User Guide*. Create the bucket in the same AWS Region where you plan to launch your Amazon EMR cluster. For example, US West (Oregon) us-west-2.

Buckets and folders that you use with Amazon EMR have the following limitations:

- Names can consist of lowercase letters, numbers, periods (.), and hyphens (-).
- Names cannot end in numbers.
- A bucket name must be unique *across all AWS accounts.*
- An output folder must be empty.

Prepare an application with input data for Amazon EMR

The most common way to prepare an application for Amazon EMR is to upload the application and its input data to Amazon S3. Then, when you submit work to your cluster you specify the Amazon S3 locations for your script and data.

In this step, you upload a sample PySpark script to your Amazon S3 bucket. We've provided a PySpark script for you to use. The script processes food establishment inspection data and returns a results file in your S3 bucket. The results file lists the top ten establishments with the most "Red" type violations.

You also upload sample input data to Amazon S3 for the PySpark script to process. The input data is a modified version of Health Department inspection results in King County, Washington, from 2006 to 2020. For more information, see King County Open Data: Food Establishment Inspection Data. Following are sample rows from the dataset.

```
name, inspection_result, inspection_closed_business, violation_type, violation_points
100 LB CLAM, Unsatisfactory, FALSE, BLUE, 5
100 PERCENT NUTRICION, Unsatisfactory, FALSE, BLUE, 5
7-ELEVEN #2361-39423A, Complete, FALSE, , 0
```

To prepare the example PySpark script for EMR

1. Copy the example code below into a new file in your editor of choice.

```
import argparse
```

```
from pyspark.sql import SparkSession

def calculate_red_violations(data_source, output_uri):
    """
    Processes sample food establishment inspection data and queries the data to find
 the top 10 establishments
    with the most Red violations from 2006 to 2020.

    :param data_source: The URI of your food establishment data CSV, such as 's3://DOC-
EXAMPLE-BUCKET/food-establishment-data.csv'.
    :param output_uri: The URI where output is written, such as 's3://DOC-EXAMPLE-
BUCKET/restaurant_violation_results'.
    """
    with SparkSession.builder.appName("Calculate Red Health Violations").getOrCreate()
 as spark:
        # Load the restaurant violation CSV data
        if data_source is not None:
            restaurants_df = spark.read.option("header", "true").csv(data_source)

        # Create an in-memory DataFrame to query
        restaurants_df.createOrReplaceTempView("restaurant_violations")

        # Create a DataFrame of the top 10 restaurants with the most Red violations
        top_red_violation_restaurants = spark.sql("""SELECT name, count(*) AS
 total_red_violations
          FROM restaurant_violations
          WHERE violation_type = 'RED'
          GROUP BY name
          ORDER BY total_red_violations DESC LIMIT 10""")

        # Write the results to the specified output URI
        top_red_violation_restaurants.write.option("header",
 "true").mode("overwrite").csv(output_uri)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '--data_source', help="The URI for you CSV restaurant data, like an S3 bucket
 location.")
    parser.add_argument(
        '--output_uri', help="The URI where output is saved, like an S3 bucket
 location.")
    args = parser.parse_args()

    calculate_red_violations(args.data_source, args.output_uri)
```

2. Save the file as `health_violations.py`.
3. Upload `health_violations.py` to Amazon S3 into the bucket you created for this tutorial. For instructions, see Uploading an object to a bucket in the *Amazon Simple Storage Service Getting Started Guide*.

To prepare the sample input data for EMR

1. Download the zip file, food_establishment_data.zip.
2. Unzip and save `food_establishment_data.zip` as `food_establishment_data.csv` on your machine.
3. Upload the CSV file to the S3 bucket that you created for this tutorial. For instructions, see Uploading an object to a bucket in the *Amazon Simple Storage Service Getting Started Guide*.

For more information about setting up data for EMR, see Prepare input data (p. 135).

Launch an Amazon EMR cluster

After you prepare a storage location and your application, you can launch a sample Amazon EMR cluster. In this step, you launch an Apache Spark cluster using the latest Amazon EMR release version.

Console

To launch a cluster with Spark installed using Quick Options

1. Open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Choose **Create cluster** to open the **Quick Options wizard**.
3. Note the default values for **Release**, **Instance type**, **Number of instances**, and **Permissions** on the **Create Cluster - Quick Options** page. These fields autofill with values that work for general-purpose clusters.
4. Enter a **Cluster name** to help you identify the cluster. For example, *`My First EMR Cluster`*.
5. Leave **Logging** enabled, but replace the **S3 folder** value with the Amazon S3 bucket you created, followed by **`/logs`**. For example, **`s3://DOC-EXAMPLE-BUCKET/logs`**. Adding **`/logs`** creates a new folder called 'logs' in your bucket, where EMR can copy the log files of your cluster.
6. Choose the **Spark** option under **Applications** to install Spark on your cluster.

   Note
   Choose the applications you want on your Amazon EMR cluster before you launch the cluster. You can't add or remove applications from a cluster after launch.
7. Choose your **EC2 key pair** under **Security and access**.
8. Choose **Create cluster** to launch the cluster and open the cluster status page.
9. Find the cluster **Status** next to the cluster name. The status changes from **Starting** to **Running** to **Waiting** as Amazon EMR provisions the cluster. You may need to choose the refresh icon on the right or refresh your browser to see status updates.

Your cluster status changes to **Waiting** when the cluster is up, running, and ready to accept work. For more information about reading the cluster summary, see View cluster status and details (p. 550). For information about cluster status, see Understanding the cluster lifecycle (p. 3).

CLI

To launch a cluster with Spark installed using the AWS CLI

1. Create IAM default roles that you can then use to create your cluster by using the following command.

   ```
   aws emr create-default-roles
   ```

   For more information about `create-default-roles`, see the AWS CLI Command Reference.
2. Create a Spark cluster with the following command. Enter a name for your cluster with the `--name` option, and specify the name of your EC2 key pair with the `--ec2-attributes` option.

   ```
   aws emr create-cluster \
   --name "<My First EMR Cluster>" \
   --release-label <emr-5.36.0> \
   --applications Name=Spark \
   --ec2-attributes KeyName=<myEMRKeyPairName> \
   --instance-type m5.xlarge \
   --instance-count 3 \
   ```

```
--use-default-roles
```

Note the other required values for `--instance-type`, `--instance-count`, and `--use-default-roles`. These values have been chosen for general-purpose clusters. For more information about `create-cluster`, see the AWS CLI Command Reference.

> **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 (^).

You should see output like the following. The output shows the `ClusterId` and `ClusterArn` of your new cluster. Note your `ClusterId`. You use the `ClusterId` to check on the cluster status and to submit work.

```
{
    "ClusterId": "myClusterId",
    "ClusterArn": "myClusterArn"
}
```

3. Check your cluster status with the following command.

```
aws emr describe-cluster --cluster-id <myClusterId>
```

You should see output like the following with the `Status` object for your new cluster.

```
{
    "Cluster": {
        "Id": "myClusterId",
        "Name": "My First EMR Cluster",
        "Status": {
            "State": "STARTING",
            "StateChangeReason": {
                "Message": "Configuring cluster software"
            }
        }
     }
}
```

The `State` value changes from `STARTING` to `RUNNING` to `WAITING` as Amazon EMR provisions the cluster.

Cluster status changes to **`WAITING`** when a cluster is up, running, and ready to accept work. For information about cluster status, see Understanding the cluster lifecycle (p. 3).

Step 2: Manage your Amazon EMR cluster

Submit work to Amazon EMR

After you launch a cluster, you can submit work to the running cluster to process and analyze data. You submit work to an Amazon EMR cluster as a *step*. A step is a unit of work made up of one or more actions. For example, you might submit a step to compute values, or to transfer and process data. You can submit steps when you create a cluster, or to a running cluster. In this part of the tutorial, you submit `health_violations.py` as a step to your running cluster. To learn more about steps, see Submit work to a cluster (p. 642).

Console

To submit a Spark application as a step using the console

1. Open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Select the name of your cluster from the **Cluster List**. The cluster state must be **Waiting**.
3. Choose **Steps**, and then choose **Add step**.
4. Configure the step according to the following guidelines:
   - For **Step type**, choose **Spark application**. You should see additional fields for **Deploy Mode**, **Spark-submit options**, and **Application location** appear.
   - For **Name**, leave the default value or type a new name. If you have many steps in a cluster, naming each step helps you keep track of them.
   - For **Deploy mode**, leave the default value **Cluster**. For more information about Spark deployment modes, see Cluster mode overview in the Apache Spark documentation.
   - Leave the **Spark-submit options** field blank. For more information about `spark-submit` options, see Launching applications with spark-submit.
   - For **Application location**, enter the location of your `health_violations.py` script in Amazon S3. For example, *`s3://DOC-EXAMPLE-BUCKET/health_violations.py`*.
   - In the **Arguments** field, enter the following arguments and values:

     ```
     --data_source s3://DOC-EXAMPLE-BUCKET/food_establishment_data.csv
     --output_uri s3://DOC-EXAMPLE-BUCKET/myOutputFolder
     ```

     Replace *`s3://DOC-EXAMPLE-BUCKET/food_establishment_data.csv`* with the S3 URI of the input data you prepared in Prepare an application with input data for Amazon EMR (p. 15).

     Replace *`DOC-EXAMPLE-BUCKET`* with the name of the bucket you created for this tutorial, and *`myOutputFolder`* with a name for your cluster output folder.
   - For **Action on failure**, accept the default option **Continue** so that if the step fails, the cluster continues to run.
5. Choose **Add** to submit the step. The step should appear in the console with a status of **Pending**.
6. Check for the step status to change from **Pending** to **Running** to **Completed**. To refresh the status in the console, choose the refresh icon to the right of the **Filter**. The script takes about one minute to run.

You will know that the step finished successfully when the status changes to **Completed**.

CLI

To submit a Spark application as a step using the AWS CLI

1. Make sure you have the `ClusterId` of the cluster you launched in Launch an Amazon EMR cluster (p. 17). You can also retrieve your cluster ID with the following command.

   ```
   aws emr list-clusters --cluster-states WAITING
   ```

2. Submit `health_violations.py` as a step with the `add-steps` command and your `ClusterId`.
   - You can specify a name for your step by replacing *`"My Spark Application"`*. In the `Args` array, replace *`s3://DOC-EXAMPLE-BUCKET/health_violations.py`* with the location of your `health_violations.py` application.

- Replace *s3://DOC-EXAMPLE-BUCKET/food_establishment_data.csv* with the S3 location of your `food_establishment_data.csv` dataset.
- Replace *s3://DOC-EXAMPLE-BUCKET/MyOutputFolder* with the S3 path of your designated bucket and a name for your cluster output folder.
- `ActionOnFailure=CONTINUE` means the cluster continues to run if the step fails.

```
aws emr add-steps \
--cluster-id <myClusterId> \
--steps Type=Spark,Name="<My Spark
 Application>",ActionOnFailure=CONTINUE,Args=[<s3://DOC-EXAMPLE-
BUCKET/health_violations.py>,--data_source,<s3://DOC-EXAMPLE-BUCKET/
food_establishment_data.csv>,--output_uri,<s3://DOC-EXAMPLE-BUCKET/MyOutputFolder>]
```

For more information about submitting steps using the CLI, see the AWS CLI Command Reference.

After you submit the step, you should see output like the following with a list of `StepIds`. Since you submitted one step, you will see just one ID in the list. Copy your step ID. You use your step ID to check the status of the step.

```
{
    "StepIds": [
        "s-1XXXXXXXXXXA"
    ]
}
```

3. Query the status of your step with the `describe-step` command.

```
aws emr describe-step --cluster-id <myClusterId> --step-id <s-1XXXXXXXXXXA>
```

You should see output like the following with information about your step.

```
{
    "Step": {
        "Id": "s-1XXXXXXXXXXA",
        "Name": "My Spark Application",
        "Config": {
            "Jar": "command-runner.jar",
            "Properties": {},
            "Args": [
                "spark-submit",
                "s3://DOC-EXAMPLE-BUCKET/health_violations.py",
                "--data_source",
                "s3://DOC-EXAMPLE-BUCKET/food_establishment_data.csv",
                "--output_uri",
                "s3://DOC-EXAMPLE-BUCKET/myOutputFolder"
            ]
        },
        "ActionOnFailure": "CONTINUE",
        "Status": {
            "State": "COMPLETED"
        }
    }
}
```

The `State` of the step changes from `PENDING` to `RUNNING` to `COMPLETED` as the step runs. The step takes about one minute to run, so you might need to check the status a few times.

You will know that the step was successful when the `State` changes to **`COMPLETED`**.

For more information about the step lifecycle, see Running steps to process data (p. 3).

View results

After a step runs successfully, you can view its output results in your Amazon S3 output folder.

To view the results of `health_violations.py`

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose the **Bucket name** and then the output folder that you specified when you submitted the step. For example, *`DOC-EXAMPLE-BUCKET`* and then *`myOutputFolder`*.
3. Verify that the following items appear in your output folder:
   - A small-sized object called _SUCCESS.
   - A CSV file starting with the prefix `part-` that contains your results.
4. Choose the object with your results, then choose **Download** to save the results to your local file system.
5. Open the results in your editor of choice. The output file lists the top ten food establishments with the most red violations. The output file also shows the total number of red violations for each establishment.

   The following is an example of `health_violations.py` results.

```
name, total_red_violations
SUBWAY, 322
T-MOBILE PARK, 315
WHOLE FOODS MARKET, 299
PCC COMMUNITY MARKETS, 251
TACO TIME, 240
MCDONALD'S, 177
THAI GINGER, 153
SAFEWAY INC #1508, 143
TAQUERIA EL RINCONSITO, 134
HIMITSU TERIYAKI, 128
```

For more information about Amazon EMR cluster output, see Configure an output location (p. 144).

(Optional) Connect to your running Amazon EMR cluster

When you use Amazon EMR, you may want to connect to a running cluster to read log files, debug the cluster, or use CLI tools like the Spark shell. Amazon EMR lets you connect to a cluster using the Secure Shell (SSH) protocol. This section covers how to configure SSH, connect to your cluster, and view log files for Spark. For more information about connecting to a cluster, see Authenticate to Amazon EMR cluster nodes (p. 447).

Authorize SSH connections to your cluster

Before you connect to your cluster, you need to modify your cluster security groups to authorize inbound SSH connections. Amazon EC2 security groups act as virtual firewalls to control inbound and outbound

traffic to your cluster. When you created your cluster for this tutorial, Amazon EMR created the following security groups on your behalf:

ElasticMapReduce-master

The default Amazon EMR managed security group associated with the master node. In an Amazon EMR cluster, the master node is an Amazon EC2 instance that manages the cluster.

ElasticMapReduce-slave

The default security group associated with core and task nodes.

To allow SSH access for trusted sources for the ElasticMapReduce-master security group with the old console

To edit your security groups, you must have permission to manage security groups for the VPC that the cluster is in. For more information, see Changing Permissions for an IAM User and the Example Policy that allows managing EC2 security groups in the *IAM User Guide*.

1. Open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Choose **Clusters**. Choose the **Name** of the cluster you want to modify.
3. Choose the **Security groups for Master** link under **Security and access**.
4. Choose **ElasticMapReduce-master** from the list.
5. Choose the **Inbound rules** tab and then **Edit inbound rules**.
6. Check for an inbound rule that allows public access with the following settings. If it exists, choose **Delete** to remove it.
   - **Type**

     SSH
   - **Port**

     22
   - **Source**

     Custom 0.0.0.0/0

     Warning
     Before December 2020, the ElasticMapReduce-master security group had a pre-configured rule to allow inbound traffic on Port 22 from all sources. This rule was created to simplify initial SSH connections to the master node. We strongly recommend that you remove this inbound rule and restrict traffic to trusted sources.
7. Scroll to the bottom of the list of rules and choose **Add Rule**.
8. For **Type**, select **SSH**.

   Selecting SSH automatically enters **TCP** for **Protocol** and **22** for **Port Range**.
9. For source, select **My IP** to automatically add your IP address as the source address. You can also add a range of **Custom** trusted client IP addresses, or create additional rules for other clients. Many network environments dynamically allocate IP addresses, so you might need to update your IP addresses for trusted clients in the future.
10. Choose **Save**.
11. Optionally, choose **ElasticMapReduce-slave** from the list and repeat the steps above to allow SSH client access to core and task nodes.

Connect to your cluster using the AWS CLI

Regardless of your operating system, you can create an SSH connection to your cluster using the AWS CLI.

To connect to your cluster and view log files using the AWS CLI

1. Use the following command to open an SSH connection to your cluster. Replace *<mykeypair.key>* with the full path and file name of your key pair file. For example, `C:\Users\<username>\.ssh\mykeypair.pem`.

```
aws emr ssh --cluster-id <j-2AL4XXXXXX5T9> --key-pair-file <~/mykeypair.key>
```

2. Navigate to `/mnt/var/log/spark` to access the Spark logs on your cluster's master node. Then view the files in that location. For a list of additional log files on the master node, see View log files on the master node (p. 564).

```
cd /mnt/var/log/spark
ls
```

Step 3: Clean up your Amazon EMR resources

Terminate your cluster

Now that you've submitted work to your cluster and viewed the results of your PySpark application, you can terminate the cluster. Terminating a cluster stops all of the cluster's associated Amazon EMR charges and Amazon EC2 instances.

When you terminate a cluster, Amazon EMR retains metadata about the cluster for two months at no charge. Archived metadata helps you clone the cluster (p. 641) for a new job or revisit the cluster configuration for reference purposes. Metadata does *not* include data that the cluster writes to S3, or data stored in HDFS on the cluster.

> **Note**
> The Amazon EMR console does not let you delete a cluster from the list view after you terminate the cluster. A terminated cluster disappears from the console when Amazon EMR clears its metadata.

Console

To terminate the cluster using the console

1. Open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Choose **Clusters**, then choose the cluster you want to terminate. For example, *`My First EMR Cluster`*.
3. Choose **Terminate** to open the **Terminate cluster** prompt.
4. Choose **Terminate** in the open prompt. Depending on the cluster configuration, termination may take 5 to 10 minutes. For more information about terminating Amazon EMR clusters, see Terminate a cluster (p. 607).

> **Note**
> If you followed the tutorial closely, termination protection should be off. Cluster termination protection prevents accidental termination. If termination protection is

on, you will see a prompt to change the setting before terminating the cluster. Choose **Change**, then **Off**.

CLI

To terminate the cluster using the AWS CLI

1. Initiate the cluster termination process with the following command. Replace *<myClusterId>* with the ID of your sample cluster. The command does not return output.

```
aws emr terminate-clusters --cluster-ids <myClusterId>
```

2. To check that the cluster termination process is in progress, check the cluster status with the following command.

```
aws emr describe-cluster --cluster-id <myClusterId>
```

Following is example output in JSON format. The cluster `Status` should change from **`TERMINATING`** to **`TERMINATED`**. Termination may take 5 to 10 minutes depending on your cluster configuration. For more information about terminating an Amazon EMR cluster, see Terminate a cluster (p. 607).

```
{
    "Cluster": {
        "Id": "j-xxxxxxxxxxxxx",
        "Name": "My Cluster Name",
        "Status": {
            "State": "TERMINATED",
            "StateChangeReason": {
                "Code": "USER_REQUEST",
                "Message": "Terminated by user request"
            }
        }
    }
}
```

Delete S3 resources

To avoid additional charges, you should delete your Amazon S3 bucket. Deleting the bucket removes all of the Amazon S3 resources for this tutorial. Your bucket should contain:

- The PySpark script
- The input dataset
- Your output results folder
- Your log files folder

You might need to take extra steps to delete stored files if you saved your PySpark script or output in a different location.

> **Note**
> Your cluster must be terminated before you delete your bucket. Otherwise, you may not be allowed to empty the bucket.

To delete your bucket, follow the instructions in How do I delete an S3 bucket? in the *Amazon Simple Storage Service User Guide*.

Next steps

You have now launched your first Amazon EMR cluster from start to finish. You have also completed essential EMR tasks like preparing and submitting big data applications, viewing results, and terminating a cluster.

Use the following topics to learn more about how you can customize your Amazon EMR workflow.

Explore big data applications for Amazon EMR

Discover and compare the big data applications you can install on a cluster in the Amazon EMR Release Guide. The Release Guide details each EMR release version and includes tips for using frameworks such as Spark and Hadoop on Amazon EMR.

Plan cluster hardware, networking, and security

In this tutorial, you created a simple EMR cluster without configuring advanced options. Advanced options let you specify Amazon EC2 instance types, cluster networking, and cluster security. For more information about planning and launching a cluster that meets your requirements, see Plan and configure clusters (p. 131) and Security in Amazon EMR (p. 353).

Manage clusters

Dive deeper into working with running clusters in Manage clusters (p. 550). To manage a cluster, you can connect to the cluster, debug steps, and track cluster activities and health. You can also adjust cluster resources in response to workload demands with EMR managed scaling (p. 610).

Use a different interface

In addition to the Amazon EMR console, you can manage Amazon EMR using the AWS Command Line Interface, the web service API, or one of the many supported AWS SDKs. For more information, see Management interfaces (p. 8).

You can also interact with applications installed on Amazon EMR clusters in many ways. Some applications like Apache Hadoop publish web interfaces that you can view. For more information, see View web interfaces hosted on Amazon EMR clusters (p. 599).

Browse the EMR technical blog

For sample walkthroughs and in-depth technical discussion of new Amazon EMR features, see the AWS big data blog.

Amazon EMR Studio

Amazon EMR Studio is a web-based integrated development environment (IDE) for fully managed Jupyter notebooks that run on Amazon EMR clusters. You can set up an EMR Studio for your team to develop, visualize, and debug applications written in R, Python, Scala, and PySpark. EMR Studio is integrated with AWS Identity and Access Management (IAM) and IAM Identity Center so users can log in using their corporate credentials.

You can create an EMR Studio at no cost. Applicable charges for Amazon S3 storage and for Amazon EMR clusters apply when you use EMR Studio. For product details and highlights, see the service page for Amazon EMR Studio.

Key features of EMR Studio

Amazon EMR Studio provides the following features:

- Authenticate users with AWS Identity and Access Management (IAM) or AWS IAM Identity Center (successor to AWS Single Sign-On) (IAM Identity Center) and your enterprise identity provider.
- Access and launch Amazon EMR clusters on demand to run Jupyter Notebook jobs.
- Connect to Amazon EMR on EKS clusters to submit work as job runs.
- Explore and save example notebooks. For more information about example notebooks, see the EMR Studio Notebook examples GitHub repository.
- Analyze data using Python, PySpark, Spark Scala, Spark R, or SparkSQL, and install custom kernels and libraries.
- Collaborate in real time with other users in the same Workspace. For more information, see Configure Workspace collaboration (p. 86).
- Use the EMR Studio SQL Explorer to browse your data catalog, run SQL queries, and download results before you work with the data in a notebook.
- Run parameterized notebooks as part of scheduled workflows using an orchestration tool such as Apache Airflow or Amazon Managed Workflows for Apache Airflow. For more information, see Orchestrating analytics jobs on EMR Notebooks using MWAA in the AWS Big Data Blog.
- Link code repositories such as GitHub and BitBucket.
- Track and debug jobs using the Spark History Server, Tez UI, or YARN timeline server.

EMR Studio is also HIPAA eligible and is certified under HITRUST CSF and SOC 2. For more information about HIPAA compliance for AWS services, see https://aws.amazon.com/compliance/hipaa-compliance/. To learn more about HITRUST CSF compliance for AWS services, see https://aws.amazon.com/compliance/hitrust/. For more information about other compliance programs for AWS services, see AWS Services in Scope by Compliance Program.

How Amazon EMR Studio works

An Amazon EMR Studio is an Amazon EMR resource that you create for a team of users. Each Studio is a self-contained, web-based integrated development environment for Jupyter notebooks that run on Amazon EMR clusters. Users log in to a Studio using corporate credentials.

Each EMR Studio that you create uses the following AWS resources:

- **An Amazon Virtual Private Cloud (VPC) with subnets** - Users run Studio kernels and applications on Amazon EMR and Amazon EMR on EKS clusters in the specified VPC. An EMR Studio can connect to any cluster in the subnets that you specify when you create the Studio.
- **IAM roles and permissions policies** - To manage user permissions, you create IAM permissions policies that you attach to a user's IAM identity or to an IAM user role. EMR Studio also uses an IAM service role and security groups to interoperate with other AWS services. For more information, see Access control (p. 29) and Define security groups to control EMR Studio network traffic (p. 73).
- **Security groups** - EMR Studio uses security groups to establish a secure network channel between the Studio and an EMR cluster.
- **An Amazon S3 backup location** - EMR Studio saves notebook work in an Amazon S3 location.

The following steps outline how to create and administer an EMR Studio:

1. Create a Studio in your AWS account with either IAM or IAM Identity Center authentication. For instructions, see Set up an Amazon EMR Studio (p. 38).
2. Assign users and groups to your Studio. Use permissions policies to set fine-grained permissions for each user. For more information, see the topic Assign and manage EMR Studio users (p. 64).
3. Start monitoring EMR Studio actions with AWS CloudTrail events. For more information, see Monitor Amazon EMR Studio actions (p. 70).
4. Provide more cluster options to Studio users with cluster templates and Amazon EMR on EKS managed endpoints.

Authentication and user login

Amazon EMR Studio supports two authentication modes: IAM authentication mode and IAM Identity Center authentication mode. IAM mode uses AWS Identity and Access Management (IAM), while IAM Identity Center mode uses AWS IAM Identity Center (successor to AWS Single Sign-On). When you create an EMR Studio, you choose the authentication mode for all users of that Studio.

IAM authentication mode

With IAM authentication mode, you can use IAM authentication or IAM federation.

IAM *authentication* lets you manage IAM identities such as users, groups, and roles in IAM. You grant users access to a Studio with IAM permissions policies and attribute-based access control (ABAC).

IAM *federation* lets you establish trust between a third-party identity provider (IdP) and AWS so that you can manage user identities through your IdP.

IAM Identity Center authentication mode

IAM Identity Center authentication mode lets you give users federated access to an EMR Studio. You can use IAM Identity Center to authenticate users and groups from your IAM Identity Center directory, your existing corporate directory, or an external IdP such as Azure Active Directory (AD). You then manage users with your identity provider (IdP).

EMR Studio supports using the following identity providers for IAM Identity Center:

- **AWS Managed Microsoft AD and self-managed Active Directory** – For more information, see Connect to your Microsoft AD directory.
- **SAML-based providers** – For a full list, see Supported identity providers.
- **The IAM Identity Center directory** – For more information, see Manage identities in IAM Identity Center.

How authentication affects login and user assignment

The authentication mode that you choose for EMR Studio affects how users log in to a Studio, how you assign a user to a Studio, and how you *authorize* (give permissions to) users to perform actions such as creating new Amazon EMR clusters.

The following table summarizes login methods for EMR Studio according to authentication mode.

EMR Studio login options by authentication mode

Authentication mode	Login method	Description
• IAM (authentication and federation) • IAM Identity Center	EMR Studio URL	Users log in to a Studio using the Studio access URL. For example, `https://xxxxxxxxxxxxxxxxxxxxxx.emrstudio-prod.us-east-1.amazonaws.com`. Users enter IAM credentials when you use IAM authentication. When you use IAM federation or IAM Identity Center, EMR Studio redirects users to your identity provider's sign-in URL to enter credentials. In the context of identity federation, this login option is called Service Provider (SP) initiated sign-in.
• IAM (federation) • IAM Identity Center	Identity provider (IdP) portal	Users log in to your identity provider's portal, such as the Azure portal, and launch the Amazon EMR console. After launching the Amazon EMR console, users select and open a Studio from the **Studios list**. You can also configure EMR Studio as a SAML application so that users can log in to a specific Studio from your identity provider's portal. For instructions, see To configure an EMR Studio as a SAML application in your IdP portal (p. 40). In the context of identity federation, this login option is called identity provider (IdP) initiated sign-in.
• IAM (authentication)	AWS Management Console	Users sign in to the AWS Management Console using IAM credentials and open a Studio from the **Studios list** in the Amazon EMR console.

The following table outlines user assignment and authorization for EMR Studio by authentication mode.

EMR Studio user assignment and authorization by authentication mode

Authentication mode	User assignment	User authorization
IAM (authentication and federation)	Allow the `CreateStudioPresignedUrl` action in an IAM permissions policy	Define IAM permissions policies that allow certain EMR Studio actions. For native IAM users, attach the IAM permissions policy to an IAM identity

Authentication mode	User assignment	User authorization
	attached to an IAM identity (user, group, or role). For federated users, allow the `CreateStudioPresignedUrl` action in an IAM in the permissions policy that you configure for the IAM role you use for federation. Use attribute-based access control (ABAC) to specify the Studio or Studios that the user can access. For instructions, see Assign a user or group to an EMR Studio (p. 64).	(user, group, or role). For federated users, allow Studio actions in the permissions policy that you configure for the IAM role you use for federation. For more information, see Configure EMR Studio user permissions (p. 47).
IAM Identity Center	Assign a user to a Studio by mapping the user to a Studio with a specified session policy. For instructions, see Assign a user or group to an EMR Studio (p. 64).	Define IAM session policies that allow certain EMR Studio actions. Map a session policy to a user when you assign the user to a Studio . For more information, see User permissions for IAM Identity Center authentication mode (p. 29).

Access control

In Amazon EMR Studio, you configure user authorization (permissions) with AWS Identity and Access Management (IAM) identity-based policies. In these policies, you specify allowed actions and resources, as well as the conditions under which the actions are allowed.

User permissions for IAM authentication mode

To set user permissions when you use IAM authentication for EMR Studio, you allow actions such as `elasticmapreduce:RunJobFlow` in an IAM permissions policy. You can create one or more permissions policies to use. For example, you might create a basic policy that does not allow a user to create new Amazon EMR clusters, and another policy that does allow cluster creation. For a list of all Studio actions, see AWS Identity and Access Management permissions for EMR Studio users (p. 49).

User permissions for IAM Identity Center authentication mode

When you use IAM Identity Center authentication, you create a single EMR Studio user role. The *user role* is a dedicated IAM role that a Studio assumes when a user logs in.

You attach IAM session policies to the EMR Studio user role. A *session policy* is a special kind of IAM permissions policy that limits what a federated user can do during a Studio login session. Session policies let you set specific permissions for a user or group without creating multiple user roles for EMR Studio.

When you assign users and groups (p. 64) to a Studio, you map a session policy to that user or group to apply fine-grained permissions. You can also update a user or group's session policy at any time. Amazon EMR stores each session policy mapping that you create.

For more information about session policies, see Policies and permissions in the *AWS Identity and Access Management User Guide*.

Workspaces

Workspaces are the primary building blocks of Amazon EMR Studio. To organize notebooks, users create one or more Workspaces in a Studio.

Similar to workspaces in JupyterLab, a Workspace preserves the state of notebook work. However, the Workspace user interface extends the open-source JupyterLab interface with additional tools to let you create and attach EMR clusters, run jobs, explore sample notebooks, and link Git repositories.

The following list includes key features of EMR Studio Workspaces:

- Workspace visibility is Studio-based. Workspaces that you create in one Studio aren't visible in other Studios.
- By default, a Workspace is shared and can be seen by all Studio users. However, only one user can open and work in a Workspace at a time. To work simultaneously with other users, you can Configure Workspace collaboration (p. 86)
- You can collaborate simultaneously with other users in a Workspace when you enable Workspace collaboration. For more information, see Configure Workspace collaboration (p. 86).
- Notebooks in a Workspace share the same EMR cluster to run commands. You can attach a Workspace to an Amazon EMR cluster running on Amazon EC2 or to an Amazon EMR on EKS virtual cluster and managed endpoint.
- Workspaces can switch over to another Availability Zone that you associate with a Studio's subnets. You can stop and restart a Workspace to prompt the failover process. When you restart a Workspace, EMR Studio launches the Workspace in a different Availability Zone in the Studio's VPC when the Studio is configured with access to multiple Availability Zones. If the Studio has only one Availability Zone, EMR Studio attempts to launch the Workspace in a different subnet. For more information, see Resolve Workspace connectivity issues (p. 86).
- A Workspace can connect to clusters in any of the subnets that are associated with a Studio.

For more information about creating and configuring EMR Studio Workspaces, see Learn Workspace basics (p. 82).

Notebook storage in Amazon EMR Studio

When you use a Workspace, EMR Studio autosaves the cells in notebook files at a regular cadence in the Amazon S3 location that is associated with your Studio. This backup process preserves work between sessions so that you can come back to it later without committing changes to a Git repository. For more information, see Save Workspace content (p. 84).

When you delete a notebook file from a Workspace, EMR Studio deletes the backup version from Amazon S3 for you. However, if you delete a Workspace without first deleting its notebook files, the notebook files remain in Amazon S3 and continue to accrue storage charges. To learn more, see Delete a Workspace and notebook files (p. 84).

Considerations and limitations

Consider the following when you work with EMR Studio:

- EMR Studio is available in the following AWS Regions: US East (N. Virginia, Ohio), US West (N. California, Oregon), Asia Pacific (Mumbai, Seoul, Singapore, Sydney, Tokyo), Canada (Central), EU (Frankfurt, Ireland, London, Paris, Stockholm), and South America (Sao Paulo).

- EMR Studio works with Amazon EMR versions 5.32.0 (EMR 5.x series) or 6.2.0 (EMR 6.x series) and later.
- To let users provision new EMR clusters running on Amazon EC2 for a Workspace, you can associate an EMR Studio with a set of cluster templates. Administrators can define cluster templates with AWS Service Catalog and can choose whether a user or group can access the cluster templates, or no cluster templates, within a Studio.
- When you define access permissions to notebook files stored in Amazon S3 or read secrets from AWS Secrets Manager, use the EMR service role. Defining these permissions using session policies isn't supported.
- You can create multiple EMR Studios to control access to EMR clusters in different VPCs.
- Use the AWS CLI to set up Amazon EMR on EKS clusters. You can then use the Studio interface to attach clusters to Workspaces with a managed endpoint to run notebook jobs.
- EMR Studio doesn't support the following Python magic commands:
  - `%alias`
  - `%alias_magic`
  - `%automagic`
  - `%macro`
  - `%%js`
  - `%%javascript`
  - Modifying `proxy_user` using `%configure`
  - Modifying `KERNEL_USERNAME` using `%env` or `%set_env`
- Amazon EMR on EKS clusters don't support SparkMagic commands for EMR Studio.
- To write multi-line Scala statements in notebook cells, make sure that all but the last line end with a period. The following example uses the correct syntax for multi-line Scala statements.

```
val df = spark.sql("SELECT * from table_name).
        filter("col1=='value'").
        limit(50)
```

Known issues

- Make sure you deactivate proxy management tools such as FoxyProxy or SwitchyOmega in the browser before you create a Studio. Active proxies can cause errors when you choose **Create Studio**, and result in a **Network Failure** error message.
- Kernels that run on Amazon EMR on EKS clusters can fail to start due to timeout issues. If you encounter an error or issue starting the kernel, close the notebook file, shut down the kernel, and then reopen the notebook file.
- The **Restart kernel** operation doesn't work as expected when you use an Amazon EMR on EKS cluster. After you select **Restart kernel**, refresh the Workspace for the restart to take effect.
- If a Workspace isn't attached to a cluster, an error message appears when a Studio user opens a notebook file and tries to select a kernel. You can ignore this error message by choosing **Ok**, but you must attach the Workspace to a cluster and select a kernel before you can run notebook code.
- When you use Amazon EMR 6.2.0 with a security configuration to set up cluster security, the Workspace interface appears blank and doesn't work as expected. We recommend that you use a different supported version of EMR if you want to configure data encryption or Amazon S3 authorization for EMRFS for a cluster. EMR Studio works with Amazon EMR versions 5.32.0 (EMR 5.x series) or 6.2.0 (EMR 6.x series) and later.
- When you launch the on-cluster Spark UI from a notebook file, you can see information about a job after running notebook code. However, when you launch the Spark History Server from the Studio **Clusters** list, the job may not appear for up to two minutes.

- When you Debug Amazon EMR running on Amazon EC2 jobs (p. 94), the links to the on-cluster Spark UI may not work or fail to appear. To regenerate the links, create a new notebook cell and run the `%%info` command.
- Jupyter Enterprise Gateway doesn't clean up idle kernels on the master node of a cluster in the following Amazon EMR release versions: 5.32.0, 5.33.0, 6.2.0, and 6.3.0. Idle kernels consume computing resources and can cause long running clusters to fail. You can configure idle kernel cleanup for Jupyter Enterprise Gateway using the following example script. You can Connect to the master node using SSH (p. 593), or submit the script as a step. For more information, see Run commands and scripts on an Amazon EMR cluster.

```
#!/bin/bash
sudo tee -a /emr/notebook-env/conf/jupyter_enterprise_gateway_config.py << EOF
c.MappingKernelManager.cull_connected = True
c.MappingKernelManager.cull_idle_timeout = 10800
c.MappingKernelManager.cull_interval = 300
EOF
sudo systemctl daemon-reload
sudo systemctl restart jupyter_enterprise_gateway
```

- When you use an auto-termination policy with Amazon EMR versions 5.32.0, 5.33.0, 6.2.0, or 6.3.0, EMR marks a cluster as idle and may automatically terminate the cluster even if you have an active Python3 kernel. This is because executing a Python3 kernel does not submit a Spark job on the cluster. To use auto-termination with a Python3 kernel, we recommend that you use Amazon EMR version 6.4.0 or later. For more information about auto-termination, see Using an auto-termination policy (p. 169).
- When you use `%%display` to display a Spark DataFrame in a table, very wide tables may get truncated. You can right-click the output and select **Create New View for Output** to get a scrollable view of the output.
- Starting a Spark-based kernel, such as PySpark, Spark, or SparkR, starts a Spark session, and running a cell in a notebook queues up Spark jobs in that session. When you interrupt a running cell, the Spark job continues to run. To stop the Spark job, you should use the on-cluster Spark UI. For instructions on how to connect to the Spark UI, see Debug applications and jobs with EMR Studio (p. 94).

Feature limitations

Amazon EMR Studio doesn't support the following Amazon EMR features:

- Attaching and running jobs on EMR clusters with a security configuration that specifies Kerberos authentication
- Clusters with multiple master nodes
- Clusters integrated with AWS Lake Formation
- Clusters using Amazon EC2 instances based on AWS Graviton2

Service limits for EMR Studio

The following table displays service limits for EMR Studio.

Item	Limit
EMR Studios	Maximum of 10 per AWS account
Subnets	Maximum of 5 associated with each EMR Studio

Item	Limit
IAM Identity Center Groups	Maximum of 5 assigned to each EMR Studio
IAM Identity Center Users	Maximum of 100 assigned to each EMR Studio

VPC and subnet best practices

Use the following best practices to set up an Amazon Virtual Private Cloud (Amazon VPC) with subnets for EMR Studio:

- You can specify a maximum of five subnets in your VPC to associate with the Studio. We recommend that you provide multiple subnets in different Availability Zones in order to support Workspace availability and give Studio users access to clusters across different Availability Zones. To learn more about working with VPCs, subnets, and Availability Zones, see VPCs and subnets in the *Amazon Virtual Private Cloud User Guide*.
- The subnets that you specify should be able to communicate with each other.
- To let users link a Workspace to publicly hosted Git repositories, you should specify only private subnets that have access to the internet through Network Address Translation (NAT). For more information about setting up a private subnet for Amazon EMR, see Private subnets (p. 308).
- When you use Amazon EMR on EKS with EMR Studio, there must be *at least one subnet in common* between your Studio and the Amazon EKS cluster that you use to register a virtual cluster. Otherwise, your managed endpoint won't appear as an option in Studio Workspaces. You can create an Amazon EKS cluster and associate it with a subnet that belongs to the Studio, or create a Studio and specify your EKS cluster's subnets.
- If you plan to use Amazon EMR on EKS with EMR Studio, choose the same VPC as your Amazon EKS cluster worker nodes.

Cluster requirements for Amazon EMR Studio

Amazon EMR Clusters Running on Amazon EC2

All Amazon EMR clusters running on Amazon EC2 that you create for an EMR Studio Workspace must meet the following requirements. Clusters that you create using the EMR Studio interface automatically meet these requirements.

- The cluster must use Amazon EMR versions 5.32.0 (EMR 5.x series) or 6.2.0 (EMR 6.x series) or later. You can create a cluster using the Amazon EMR console, AWS Command Line Interface, or SDK, and then attach it to an EMR Studio Workspace. Studio users can also provision and attach clusters when creating or working in an Amazon EMR Workspace. For more information, see Attach a cluster to a Workspace (p. 89).
- The cluster must be within an Amazon Virtual Private Cloud. The EC2-Classic platform isn't supported.
- The cluster must have Spark, Livy, and Jupyter Enterprise Gateway installed. If you plan to use the cluster for SQL Explorer, you should install both Presto and Spark.
- To use SQL Explorer, the cluster must use Amazon EMR version 5.34.0 or later or version 6.4.0 or later and have Presto installed. If you want to specify the AWS Glue Data Catalog as the Hive metastore for Presto, you must configure it on the cluster. For more information, see Using Presto with the AWS Glue Data Catalog.
- The cluster must be in a private subnet with network address translation (NAT) to use publicly-hosted Git repositories with EMR Studio.

We recommend the following cluster configurations when you work with EMR Studio.

- Set deploy mode for Spark sessions to cluster mode. Cluster mode places the application master processes on the core nodes and not on the master node of a cluster. Doing so relieves the master node of potential memory pressures. For more information, see Cluster Mode Overview in the Apache Spark documentation.
- Change the Livy timeout from the default of one hour to six hours as in the following example configuration.

```
{
    "classification":"livy-conf",
        "Properties":{
            "livy.server.session.timeout":"6h",
            "livy.spark.deploy-mode":"cluster"
        }
}
```

- Create diverse instance fleets with up to 30 instances, and select multiple instance types in your Spot Instance fleet. For example, you might specify the following memory-optimized instance types for Spark workloads: r5.2x, r5.4x, r5.8x, r5.12x, r5.16x, r4.2x, r4.4x, r4.8x, r4.12, etc. For more information, see Configure instance fleets (p. 316).
- Use the capacity-optimized allocation strategy for Spot Instances to help Amazon EMR make effective instance selections based on real-time capacity insights from Amazon EC2. For more information, see Allocation strategy for instance fleets (p. 319).
- Enable managed scaling on your cluster. Set the maximum core nodes parameter to the minimum persistent capacity that you plan to use, and configure scaling on a well-diversified task fleet that runs on Spot Instances to save on costs. For more information, see Using EMR managed scaling in Amazon EMR (p. 610).

We also urge you to keep Amazon EMR Block Public Access enabled, and that to restrict inbound SSH traffic to trusted sources. Inbound access to a cluster lets users run notebooks on the cluster. For more information, see Using Amazon EMR block public access (p. 542) and Control network traffic with security groups (p. 532).

Amazon EMR on EKS Clusters

In addition to EMR clusters running on Amazon EC2, you can set up and manage Amazon EMR on EKS clusters for EMR Studio using the AWS CLI. Set up EMR on EKS clusters using the following guidelines:

- Create a managed HTTPS endpoint for the Amazon EMR on EKS cluster. Users attach a Workspace to a managed endpoint. The Amazon Elastic Kubernetes Service (EKS) cluster that you use to register a virtual cluster must have a private subnet to support managed endpoints.
- Use an Amazon EKS cluster with at least one private subnet and network address translation (NAT) when you want to use publicly-hosted Git repositories.
- Avoid using Amazon EKS optimized Arm Amazon Linux AMIs, which aren't supported for EMR on EKS managed endpoints.
- Avoid using AWS Fargate-only Amazon EKS clusters, which aren't supported.

Configure Amazon EMR Studio

This section is for EMR Studio administrators. It covers how to set up an EMR Studio for your team and provides instructions for tasks such as assigning users and groups, setting up cluster templates, and optimizing Apache Spark for EMR Studio.

Topics

- Administrator permissions to create and manage an EMR Studio (p. 35)
- Set up an Amazon EMR Studio (p. 38)

- Manage an Amazon EMR Studio (p. 69)
- Define security groups to control EMR Studio network traffic (p. 73)
- Create AWS CloudFormation templates for Amazon EMR Studio (p. 75)
- Establish access and permissions for Git-based repositories (p. 79)
- Optimize Spark jobs in EMR Studio (p. 81)

Administrator permissions to create and manage an EMR Studio

The IAM permissions described on this page permit you to create and manage an EMR Studio. For detailed information about each required permission, see Permissions required to manage an EMR Studio (p. 35).

Permissions required to manage an EMR Studio

The following table lists the operations related to creating and managing an EMR Studio. The table also displays the permissions needed for each operation.

> **Note**
> You only need IAM Identity Center and Studio `SessionMapping` actions when you use IAM Identity Center authentication mode.

Permissions to create and manage an EMR Studio

Operation	Permissions

Create a Studio	`"elasticmapreduce:CreateStudio",`
	`"sso:CreateManagedApplicationInstance",`
	`"iam:PassRole"`

Describe a Studio	`"elasticmapreduce:DescribeStudio",` `"sso:GetManagedApplicationInstance"`
List Studios	`"elasticmapreduce:ListStudios"`
Delete a Studio	`"elasticmapreduce:DeleteStudio",` `"sso:DeleteManagedApplicationInstance"`
Additional permissions required when you use IAM Identity Center mode	

Assign users or groups to a Studio	`"elasticmapreduce:CreateStudioSessionMapping",`
	`"sso:GetProfile",`
	`"sso:ListDirectoryAssociations",`
	`"sso:ListProfiles",`
	`"sso:AssociateProfile"`
	`"sso-directory:SearchUsers",`
	`"sso-directory:SearchGroups",`
	`"sso-directory:DescribeUser",`
	`"sso-directory:DescribeGroup"`

Retrieve Studio assignment details for a specific user or group	`"sso-directory:SearchUsers",`
	`"sso-directory:SearchGroups",`
	`"sso-directory:DescribeUser",`

Operation	Permissions

	`"sso-directory:DescribeGroup",`
	`"sso:GetManagedApplicationInstance",`
	`"elasticmapreduce:GetStudioSessionMapping"`

List all users and groups assigned to a Studio	`"elasticmapreduce:ListStudioSessionMappings"`

Update the session policy attached to a user or group assigned to a Studio	`"sso-directory:SearchUsers",`
	`"sso-directory:SearchGroups",`
	`"sso-directory:DescribeUser",`
	`"sso-directory:DescribeGroup",`
	`"sso:GetManagedApplicationInstance",`
	`"elasticmapreduce:UpdateStudioSessionMapping"`

Remove a user or group from a Studio	`"elasticmapreduce:DeleteStudioSessionMapping",`
	`"sso-directory:SearchUsers",`
	`"sso-directory:SearchGroups",`
	`"sso-directory:DescribeUser",`
	`"sso-directory:DescribeGroup",`
	`"sso:ListDirectoryAssociations",`
	`"sso:GetProfile",`
	`"sso:GetManagedApplicationInstance"`
	`"sso:ListProfiles",`
	`"sso:DisassociateProfile"`

To create a policy with admin permissions for EMR Studio

1. Follow the instructions in Creating IAM policies to create a policy using one of the following examples. The permissions you need depend on your authentication mode for EMR Studio (p. 38).

   Insert your own values for these items:

   - Replace *<your-resource-ARN>* to specify the Amazon Resource Name (ARN) of the object or objects that the statement covers for your use cases.
   - Replace *<region>* with the code of the AWS Region where you plan to create the Studio.
   - Replace *<aws-account_id>* with the ID of the AWS account for the Studio.
   - Replace *<EMRStudio-Service-Role>* and *<EMRStudio-User-Role>* with the names of your EMR Studio service role (p. 42) and EMR Studio user role (p. 47).

   Example Example policy: Admin permissions when you use IAM authentication mode

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Resource": "arn:aws:elasticmapreduce:<region>:<aws-account-id>:studio/*",
            "Action": [
                "elasticmapreduce:CreateStudio",
                "elasticmapreduce:DescribeStudio",
                "elasticmapreduce:DeleteStudio"
            ]
        },
        {
            "Effect": "Allow",
```

```
            "Resource": "<your-resource-ARN>",
            "Action": [
                "elasticmapreduce:ListStudios"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iam::<aws-account-id>:role/<EMRStudio-Service-Role>"
            ],
            "Action": "iam:PassRole"
        }
    ]
}
```

Example Example policy: Admin permissions when you use IAM Identity Center authentication mode

Note

IAM Identity Center and IAM Identity Center Directory APIs don't support specifying an ARN in the resource element of an IAM policy statement. To allow access to IAM Identity Center and IAM Identity Center Directory, the following permissions specify all resources, "Resource":"*", for IAM Identity Center actions. For more information, see Actions, resources, and condition keys for IAM Identity Center Directory.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Resource": "arn:aws:elasticmapreduce:<region>:<aws-account-id>:studio/*",
            "Action": [
                "elasticmapreduce:CreateStudio",
                "elasticmapreduce:DescribeStudio",
                "elasticmapreduce:DeleteStudio",
                "elasticmapreduce:CreateStudioSessionMapping",
                "elasticmapreduce:GetStudioSessionMapping",
                "elasticmapreduce:UpdateStudioSessionMapping",
                "elasticmapreduce:DeleteStudioSessionMapping"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": "<your-resource-ARN>",
            "Action": [
                "elasticmapreduce:ListStudios",
                "elasticmapreduce:ListStudioSessionMappings"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iam::<aws-account-id>:role/<EMRStudio-Service-Role>",
                "arn:aws:iam::<aws-account-id>:role/<EMRStudio-User-Role>"
            ],
            "Action": "iam:PassRole"
        },
        {
            "Effect": "Allow",
            "Resource": "*",
            "Action": [
                "sso:CreateManagedApplicationInstance",
                "sso:GetManagedApplicationInstance",
```

```
                "sso:DeleteManagedApplicationInstance",
                "sso:AssociateProfile",
                "sso:DisassociateProfile",
                "sso:GetProfile",
                "sso:ListDirectoryAssociations",
                "sso:ListProfiles",
                "sso-directory:SearchUsers",
                "sso-directory:SearchGroups",
                "sso-directory:DescribeUser",
                "sso-directory:DescribeGroup"
            ]
        }
    ]
}
```

2. Attach the policy to your IAM identity (user, role, or group). For instructions, see Adding and removing IAM identity permissions.

Set up an Amazon EMR Studio

Complete the following steps to set up an Amazon EMR Studio.

Before you start

> **Note**
> If you plan to use EMR Studio with Amazon EMR on EKS, we recommend that you first set up Amazon EMR on EKS for EMR Studio before you set up a Studio.

Before you set up an EMR Studio, make sure you have the following items:

- An AWS account. For instructions, see Setting up Amazon EMR (p. 12).
- Permissions to create and manage an EMR Studio. For more information, see the section called "Administrator permissions to create an EMR Studio" (p. 35).
- An Amazon S3 bucket where EMR Studio can back up the Workspaces and notebook files in your Studio. For instructions, see Creating a bucket in the *Amazon Simple Storage Service (S3) User Guide*.
- An Amazon Virtual Private Cloud (VPC) for the Studio, and a maximum of five subnets. For tips on how to configure networking, see VPC and subnet best practices (p. 33).

To set up an EMR Studio

1. Choose an authentication mode for Amazon EMR Studio (p. 38)
2. Create the following Studio resources.
   - Create an EMR Studio service role (p. 42)
   - Configure EMR Studio user permissions (p. 47)
   - (Optional) Define security groups to control EMR Studio network traffic (p. 73).
3. Create an EMR Studio (p. 60)
4. Assign a user or group to an EMR Studio (p. 64)

After you complete the setup steps, you can Use an Amazon EMR Studio (p. 82).

Choose an authentication mode for Amazon EMR Studio

EMR Studio supports two authentication modes: IAM authentication mode and IAM Identity Center authentication mode. IAM mode uses AWS Identity and Access Management (IAM), while IAM Identity Center mode uses AWS IAM Identity Center (successor to AWS Single Sign-On). When you create an EMR

Studio, you choose the authentication mode for all users of that Studio. For more information about the different authentication modes, see Authentication and user login (p. 27).

Use the following table to choose an authentication mode for EMR Studio.

If you are...	We recommend...
Already familiar with or have previously set up IAM authentication or federation	IAM authentication mode (p. 39), which offers the following benefits: • Provides quick setup for EMR Studio if you already manage identities such as users and groups in IAM. • Works with identity providers that are compatible with OpenID Connect (OIDC) or Security Assertion Markup Language 2.0 (SAML 2.0). • Supports using multiple identity providers with the same AWS account. • Available in a wide number of AWS Regions. • Compliant with SOC 2.
New to AWS or Amazon EMR	IAM Identity Center authentication mode (p. 40), which provides the following features: • Supports easy user and group assignment to AWS resources. • Works with Microsoft Active Directory and SAML 2.0 identity providers. • Facilitates multi-account federation setup so that you don't have to separately configure federation for each AWS account in your organization.

Set up IAM authentication mode for Amazon EMR Studio

With IAM authentication mode, you can use either IAM authentication or IAM federation. IAM *authentication* lets you manage IAM identities such as users, groups, and roles in IAM. You grant users access to a Studio with IAM permissions policies and attribute-based access control (ABAC). IAM *federation* lets you establish trust between a third-party identity provider (IdP) and AWS so that you can manage user identities through your IdP.

> **Note**
> If you already use IAM to control access to AWS resources, or if you've already configured your identity provider (IdP) for IAM, see User permissions for IAM authentication mode (p. 29) to set user permissions when you use IAM authentication mode for EMR Studio.

Use IAM federation for Amazon EMR Studio

To use IAM federation for EMR Studio, you create a trust relationship between your AWS account and your identity provider (IdP) and enable federated users to access the AWS Management Console. The steps you take to create this trust relationship differ depending on your IdP's federation standard.

In general, you complete the following tasks to configure federation with an external IdP. For complete instructions, see Enabling SAML 2.0 federated users to access the AWS Management Console and

Enabling custom identity broker access to the AWS Management Console in the *AWS Identity and Access Management User Guide*.

1. Gather information from your IdP. This usually means generating a metadata document to validate SAML authentication requests from your IdP.
2. Create an identity provider IAM entity to store information about your IdP. For instructions, see Creating IAM identity providers.
3. Create one or more IAM roles for your IdP. EMR Studio assigns a role to a federated user when the user logs in. The role permits your IdP to request temporary security credentials for access to AWS. For instructions, see Creating a role for a third-party identity provider (federation). The permissions policies that you assign to the role determine what federated users can do in AWS and in an EMR Studio. For more information, see User permissions for IAM authentication mode (p. 29).
4. (For SAML providers) Complete the SAML trust by configuring your IdP with information about AWS and the roles that you want federated users to assume. This configuration process creates *relying party trust* between your IdP and AWS. For more information, see Configuring your SAML 2.0 IdP with relying party trust and adding claims.

To configure an EMR Studio as a SAML application in your IdP portal

You can configure a particular EMR Studio as a SAML application using a deep link to the Studio. Doing so lets users log in to your IdP portal and launch a specific Studio instead of navigating through the Amazon EMR console.

- Use the following format to configure a deep link to your EMR Studio as a landing URL after SAML assertion verification.

```
https://console.aws.amazon.com/emr/home?region=<aws-region>#studio/<your-studio-id>/
start
```

Set up IAM Identity Center authentication mode for Amazon EMR Studio

To prepare AWS IAM Identity Center (successor to AWS Single Sign-On) for EMR Studio, you must configure your identity source and provision users and groups. Provisioning is the process of making user and group information available for use by IAM Identity Center and by applications that use IAM Identity Center. For more information, see User and group provisioning.

EMR Studio supports using the following identity providers for IAM Identity Center:

- **AWS Managed Microsoft AD and self-managed Active Directory** – For more information, see Connect to your Microsoft AD directory.
- **SAML-based providers** – For a full list, see Supported identity providers.
- **The IAM Identity Center directory** – For more information, see Manage identities in IAM Identity Center.

To set up IAM Identity Center for EMR Studio

1. To set up IAM Identity Center for EMR Studio, you need the following:
   - A management account in your AWS organization if you use multiple accounts in your organization.

     Note
     You should only use your management account to enable IAM Identity Center and *provision* users and groups. After you set up IAM Identity Center, use a member account

to create an EMR Studio and *assign* users and groups. To learn more about AWS terminology, see AWS Organizations terminology and concepts.

- If you enabled IAM Identity Center before November 25, 2019, you might have to enable applications that use IAM Identity Center for the accounts in your AWS organization. For more information, see Enable IAM Identity Center-integrated applications in AWS accounts.
- Make sure that you have the prerequisites listed on the IAM Identity Center prerequisites page.

2. Follow the instructions in Enable IAM Identity Center to enable IAM Identity Center in the AWS Region where you want to create the EMR Studio.
3. Connect IAM Identity Center to your identity provider and provision the users and groups that you want to assign to the Studio.

If you use...	Do this...
A Microsoft AD Directory	1. Follow the instructions in Connect to your Microsoft AD directory to connect your self-managed Active Directory or AWS Managed Microsoft AD directory using AWS Directory Service. 2. To provision users and groups for IAM Identity Center, you can sync identity data from your source AD to IAM Identity Center. You can sync identities from your source AD in many ways. One way is to assign AD users or groups to an AWS account in your organization. For instructions, see Single sign-on. Synchronization can take up to two hours. After you complete this step, synced users and groups appear in your Identity Store. **Note** Users and groups don't appear in your Identity Store until you synchronize user and group information or use just-in-time (JIT) user provisioning. For more information, see Provisioning when users come from Active Directory. 3. (Optional) After you sync AD users and groups, you can remove their access to your AWS account that you configured in the previous step. For instructions, see Remove user access.
An external identity provider	Follow the instructions in Connect to your external identity provider.
The IAM Identity Center directory	When you create users and groups in IAM Identity Center, provisioning is automatic. For more information, see Manage identities in IAM Identity Center.

You can now assign users and groups from your Identity Store to an EMR Studio. For instructions, see Assign a user or group to an EMR Studio (p. 64).

Create an EMR Studio service role

About the EMR Studio service role

Each EMR Studio uses an IAM role with permissions that let the Studio interact with other AWS services. This service role must include permissions that allow EMR Studio to establish a secure network channel between Workspaces and clusters, to store notebook files in Amazon S3 Control, and to access the AWS Secrets Manager while linking a Workspace to a Git repository.

Use the Studio service role (instead of session policies) to define all Amazon S3 access permissions for storing notebook files, and to define AWS Secrets Manager access permissions.

How to create a service role for EMR Studio

1. Follow the instructions in Creating a role to delegate permissions to an AWS service to create the service role using the following trust policy.

   Important
   The following trust policy includes the `aws:SourceArn` and `aws:SourceAccount` global condition keys to limit the permissions that you give EMR Studio to particular resources in your account. Doing so can protect you against the confused deputy problem.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "elasticmapreduce.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "<account-id>"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:elasticmapreduce:<region>:<account-id>:*"
        }
      }
    }
  ]
}
```

2. Remove the default role permissions. Then, include the permissions from the following sample IAM permissions policy. Alternatively, you can create a custom policy that uses the EMR Studio service role permissions (p. 46).

   Where applicable, change `"Resource":"*"` in the following policy to specify the Amazon Resource Name (ARN) of the resource or resources that the statement covers for your use cases.

   Important

   - Access for the `ModifyNetworkInterfaceAttribute` API must remain as-is in the following policy due to technical limitations with Amazon EC2 tag-based access control and the way EMR Studio uses `ModifyNetworkInterfaceAttribute`.
   - The following statements must remain unchanged in order for EMR Studio to work with the service role: `AllowAddingEMRTagsDuringDefaultSecurityGroupCreation` and `AllowAddingTagsDuringEC2ENICreation`.
   - To use the example policy, you must tag the following resources with the key "**`for-use-with-amazon-emr-managed-policies`**" and value "**`true`**".

- Your Amazon Virtual Private Cloud (VPC) for EMR Studio.
- Each subnet that you want to use with the Studio.
- Any custom EMR Studio security groups. You must tag any security groups that you created during the EMR Studio preview period if you want to continue to use them.
- Secrets maintained in AWS Secrets Manager that Studio users use to link Git repositories to a Workspace.

You can apply tags to resources using the **Tags** tab on the relevant resource screen in the AWS Management Console.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowEMRReadOnlyActions",
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:ListInstances",
        "elasticmapreduce:DescribeCluster",
        "elasticmapreduce:ListSteps"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowEC2ENIActionsWithEMRTags",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterfacePermission",
        "ec2:DeleteNetworkInterface"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:network-interface/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
        }
      }
    },
    {
      "Sid": "AllowEC2ENIAttributeAction",
      "Effect": "Allow",
      "Action": [
        "ec2:ModifyNetworkInterfaceAttribute"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:instance/*",
        "arn:aws:ec2:*:*:network-interface/*",
        "arn:aws:ec2:*:*:security-group/*"
      ]
    },
    {
      "Sid": "AllowEC2SecurityGroupActionsWithEMRTags",
      "Effect": "Allow",
      "Action": [
        "ec2:AuthorizeSecurityGroupEgress",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:RevokeSecurityGroupEgress",
        "ec2:RevokeSecurityGroupIngress",
        "ec2:DeleteNetworkInterfacePermission"
      ],
      "Resource": "*",
```

```
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
        }
      }
    },
    {
      "Sid": "AllowDefaultEC2SecurityGroupsCreationWithEMRTags",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateSecurityGroup"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:security-group/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/for-use-with-amazon-emr-managed-policies": "true"
        }
      }
    },
    {
      "Sid": "AllowDefaultEC2SecurityGroupsCreationInVPCWithEMRTags",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateSecurityGroup"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:vpc/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
        }
      }
    },
    {
      "Sid": "AllowAddingEMRTagsDuringDefaultSecurityGroupCreation",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateTags"
      ],
      "Resource": "arn:aws:ec2:*:*:security-group/*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/for-use-with-amazon-emr-managed-policies": "true",
          "ec2:CreateAction": "CreateSecurityGroup"
        }
      }
    },
    {
      "Sid": "AllowEC2ENICreationWithEMRTags",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:network-interface/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/for-use-with-amazon-emr-managed-policies": "true"
        }
      }
    },
    {
```

```
      "Sid": "AllowEC2ENICreationInSubnetAndSecurityGroupWithEMRTags",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:subnet/*",
        "arn:aws:ec2:*:*:security-group/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
        }
      }
    },
    {
      "Sid": "AllowAddingTagsDuringEC2ENICreation",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateTags"
      ],
      "Resource": "arn:aws:ec2:*:*:network-interface/*",
      "Condition": {
        "StringEquals": {
          "ec2:CreateAction": "CreateNetworkInterface"
        }
      }
    },
    {
      "Sid": "AllowEC2ReadOnlyActions",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeTags",
        "ec2:DescribeInstances",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowSecretsManagerReadOnlyActionsWithEMRTags",
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": "arn:aws:secretsmanager:*:*:secret:*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
        }
      }
    },
    {
      "Sid": "AllowWorkspaceCollaboration",
      "Effect": "Allow",
      "Action": [
        "iam:GetUser",
        "iam:GetRole",
        "iam:ListUsers",
        "iam:ListRoles",
        "sso:GetManagedApplicationInstance",
        "sso-directory:SearchUsers"
      ],
      "Resource": "*"
```

```
    }
  ]
}
```

3. Give your service role read and write access to your Amazon S3 location for EMR Studio. Use the following minimum set of permissions. For more information, see the Amazon S3: Allows read and write access to objects in an S3 Bucket, programmatically and in the console example.

```
"s3:PutObject",
"s3:GetObject",
"s3:GetEncryptionConfiguration",
"s3:ListBucket",
"s3:DeleteObject"
```

If you encrypt your Amazon S3 bucket, include the following permissions for AWS Key Management Service.

```
"kms:Decrypt",
"kms:GenerateDataKey",
"kms:ReEncryptFrom",
"kms:ReEncryptTo",
"kms:DescribeKey"
```

EMR Studio service role permissions

The following table lists the operations that EMR Studio performs using the service role, along with the IAM actions required for each operation.

Operation	Actions

Establish a secure network channel between a Workspace and an EMR cluster, and perform necessary cleanup actions.	`"ec2:CreateNetworkInterface",`
	`"ec2:CreateNetworkInterfacePermission",`
	`"ec2:DeleteNetworkInterface",`
	`"ec2:DeleteNetworkInterfacePermission",`
	`"ec2:DescribeNetworkInterfaces",`
	`"ec2:ModifyNetworkInterfaceAttribute",`
	`"ec2:AuthorizeSecurityGroupEgress",`
	`"ec2:AuthorizeSecurityGroupIngress",`
	`"ec2:CreateSecurityGroup",`
	`"ec2:DescribeSecurityGroups",`
	`"ec2:RevokeSecurityGroupEgress",`
	`"ec2:DescribeTags",`
	`"ec2:DescribeInstances",`
	`"ec2:DescribeSubnets",`
	`"ec2:DescribeVpcs",`
	`"elasticmapreduce:ListInstances",`
	`"elasticmapreduce:DescribeCluster",`
	`"elasticmapreduce:ListSteps"`

Use Git credentials stored in AWS Secrets Manager to link Git repositories to a Workspace.	`"secretsmanager:GetSecretValue"`
Apply AWS tags to the network interface and default security groups that EMR Studio creates while setting up the secure network channel. For	`"ec2:CreateTags"`

Operation	Actions
more information, see Tagging AWS resources.	

Access or upload notebook files and metadata to Amazon S3.	`"s3:PutObject",`
	`"s3:GetObject",`
	`"s3:GetEncryptionConfiguration",`
	`"s3:ListBucket",`
	`"s3:DeleteObject"`
	
	If you use an encrypted Amazon S3 bucket, include the following permissions.
	
	`"kms:Decrypt",`
	`"kms:GenerateDataKey",`
	`"kms:ReEncryptFrom",`
	`"kms:ReEncryptTo",`
	`"kms:DescribeKey"`

Enable and configure Workspace collaboration.	`"iam:GetUser",`
	`"iam:GetRole",`
	`"iam:ListUsers",`
	`"iam:ListRoles",`
	`"sso:GetManagedApplicationInstance",`
	`"sso-directory:SearchUsers"`

Configure EMR Studio user permissions

You must configure user permissions policies for Amazon EMR Studio so that you can set fine-grained user and group permissions. For information about how user permissions work in EMR Studio, see Access control (p. 29) in How Amazon EMR Studio works (p. 26).

> **Note**
> The permissions covered in this section don't enforce data access control. To manage access to input datasets, you should configure permissions for the clusters that your Studio uses. For more information, see Security in Amazon EMR (p. 353).

Create an EMR Studio user role for IAM Identity Center authentication mode

You must create an EMR Studio user role when you use IAM Identity Center authentication mode.

To create a user role for EMR Studio

1. Follow the instructions in Creating a role to delegate permissions to an AWS service in the *AWS Identity and Access Management User Guide* to create a user role.

   Use the following trust relationship policy when you create the role.

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "elasticmapreduce.amazonaws.com"
      },
```

```
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. Remove the default role permissions and policies.
3. Attach your EMR Studio session policies to the user role before you assign users and groups to a Studio. For instructions on how to create session policies, see Create permissions policies for EMR Studio users (p. 48).

Create permissions policies for EMR Studio users

Complete the following steps to create permissions policies for EMR Studio.

> **Note**
> To set Amazon S3 access permissions for storing notebook files and AWS Secrets Manager access permissions to read secrets while linking Workspaces to Git repositories, use the EMR Studio service role.

1. Create one or more IAM permissions policy that specify which actions a user can take in your Studio. For example, you can create three separate policies for basic, intermediate, and advanced Studio users with the example policies on this page.

   The AWS Identity and Access Management permissions for EMR Studio users (p. 49) table breaks down each Studio operation that a user might perform and lists the minimum IAM actions required to perform that operation. For instructions, see Creating IAM policies.

   Your permissions policy must include the following statements.

```
{
            "Sid": "AllowAddingTagsOnSecretsWithEMRStudioPrefix",
            "Effect": "Allow",
            "Action": "secretsmanager:TagResource",
            "Resource": "arn:aws:secretsmanager:*:*:secret:emr-studio-*"
},
{
            "Sid": "AllowPassingServiceRoleForWorkspaceCreation",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::*:role/<your-emr-studio-service-role>"
            ],
            "Effect": "Allow"
}
```

   Set ownership for Workspace collaboration

   Workspace collaboration lets multiple users work simultaneously in the same Workspace and can be configured with the **Collaboration** panel in the Workspace UI. In order to see and use the **Collaboration** panel, a user must have the following permissions. Any user with these permissions can see and use the **Collaboration** panel.

```
"elasticmapreduce:UpdateEditor",
"elasticmapreduce:PutWorkspaceAccess",
"elasticmapreduce:DeleteWorkspaceAccess",
"elasticmapreduce:ListWorkspaceAccessIdentities"
```

   To restrict access to the **Collaboration** panel, you can use tag-based access control. When a user creates a Workspace, EMR Studio applies a default tag with a key of `creatorUserId` whose value is the ID of the user creating the Workspace.

Note
EMR Studio did not add the `creatorUserId` tag to Workspaces that were created before November 16, 2021. To restrict who can configure collaboration, we recommend that you manually add the `creatorUserId` tag to your Workspace and then use tag-based access control in your user permissions policies.

The following example statement allows a user to configure collaboration for any Workspace with the tag key `creatorUserId` whose value matches the user's ID (indicated by the policy variable `aws:userId`). In other words, the statement lets a user configure collaboration for the Workspaces that they create. To learn more about policy variables, see IAM policy elements: Variables and tags.

```
{
    "Sid": "UserRolePermissionsForCollaboration",
    "Action": [
        "elasticmapreduce:UpdateEditor",
        "elasticmapreduce:PutWorkspaceAccess",
        "elasticmapreduce:DeleteWorkspaceAccess",
        "elasticmapreduce:ListWorkspaceAccessIdentities"
    ],
    "Resource": "*",
    "Effect": "Allow",
    "Condition": {
        "StringEquals": {
            "elasticmapreduce:ResourceTag/creatorUserId": "${aws:userid}"
        }
    }
}
```

2. Attach the permissions policy to your IAM identity.

   The following table summarizes which IAM identity you attach a permissions policy to, depending on your EMR Studio authentication mode. For instructions on how to attach a policy, see Adding and removing IAM identity permissions.

If you use...	Attach the policy to...
IAM authentication	Your IAM identities (users, groups of users, or roles). For example, you can attach a permissions policy to a user in your AWS account.
IAM federation with an external identity provider (IdP)	The IAM role or roles that you create for your external IdP. For example, an IAM for SAML 2.0 federation. EMR Studio uses the permissions that you attach to your IAM role(s) for users with federated access to a Studio.
IAM Identity Center	Your Amazon EMR Studio user role.

AWS Identity and Access Management permissions for EMR Studio users

The following table includes each Amazon EMR Studio operation that a user might perform, and lists the minimum IAM actions needed to perform that operation. You allow these actions in your IAM permissions policies (when you use IAM authentication) or in your session policies (when you use IAM Identity Center authentication) for EMR Studio.

The table also displays the operations allowed in each of example permissions policy for EMR Studio. For more information about the example permissions policies, see Create permissions policies for EMR Studio users (p. 48).

Action	Basic	Intermediate	Advanced	Associated actions
Create and delete Workspaces	Yes	Yes	Yes	`"elasticmapreduce:CreateEditor", "elasticmapreduce:DescribeEditor", "elasticmapreduce:ListEditors", "elasticmapreduce:DeleteEditor"`
View the Collaboration panel, enable Workspace collaboration, and add collaborators. For more information, see Set ownership for Workspace collaboration (p. 48).	Yes	Yes	Yes	`"elasticmapreduce:UpdateEditor", "elasticmapreduce:PutWorkspaceAccess", "elasticmapreduce:DeleteWorkspaceAccess", "elasticmapreduce:ListWorkspaceAccessIdentiti`
See a list of Amazon S3 Control storage buckets in the same account as the Studio when creating a new EMR cluster, and access container logs when using a web UI to debug applications	Yes	Yes	Yes	`"s3:ListAllMyBuckets", "s3:ListBucket", "s3:GetBucketLocation", "s3:GetObject"`
Access Workspaces	Yes	Yes	Yes	`"elasticmapreduce:DescribeEditor", "elasticmapreduce:ListEditors", "elasticmapreduce:StartEditor", "elasticmapreduce:StopEditor", "elasticmapreduce:OpenEditorInConsole"`
Attach or detach existing Amazon EMR clusters associated with the Workspace	Yes	Yes	Yes	`"elasticmapreduce:AttachEditor", "elasticmapreduce:DetachEditor", "elasticmapreduce:ListClusters", "elasticmapreduce:DescribeCluster", "elasticmapreduce:ListInstanceGroups", "elasticmapreduce:ListBootstrapActions"`
Attach or detach Amazon EMR on EKS clusters	Yes	Yes	Yes	`"elasticmapreduce:AttachEditor", "elasticmapreduce:DetachEditor", "emr-containers:ListVirtualClusters", "emr-containers:DescribeVirtualCluster", "emr-containers:ListManagedEndpoints",`

Action	Basic	Intermediate	Advanced	Associated actions
				`"emr-containers:DescribeManagedEndpoint",` `"emr-containers:CreateAccessTokenForManagedEndpoin`

Debug Amazon EMR on EC2 jobs with persistent application user interfaces	Yes	Yes	Yes	`"elasticmapreduce:CreatePersistentAppUI",`
				`"elasticmapreduce:DescribePersistentAppUI",`
				`"elasticmapreduce:GetPersistentAppUIPresigned`
				`"elasticmapreduce:ListClusters",`
				`"elasticmapreduce:ListSteps",`
				`"elasticmapreduce:DescribeCluster",`
				`"s3:ListBucket",`
				`"s3:GetObject"`

Debug Amazon EMR on EC2 jobs with on-cluster application user interfaces	Yes	Yes	Yes	`"elasticmapreduce:GetOnClusterAppUIPresignedU`

Debug Amazon EMR on EKS job runs using the Spark History Server	Yes	Yes	Yes	`"elasticmapreduce:CreatePersistentAppUI",`
				`"elasticmapreduce:DescribePersistentAppUI",`
				`"elasticmapreduce:GetPersistentAppUIPresigned`
				`"emr-containers:ListVirtualClusters",`
				`"emr-containers:DescribeVirtualCluster",`
				`"emr-containers:ListJobRuns",`
				`"emr-containers:DescribeJobRun",`
				`"s3:ListBucket",`
				`"s3:GetObject"`

Create and delete Git repositories	Yes	Yes	Yes	`"elasticmapreduce:CreateRepository",`
				`"elasticmapreduce:DeleteRepository",`
				`"elasticmapreduce:ListRepositories",`
				`"elasticmapreduce:DescribeRepository",`
				`"secretsmanager:CreateSecret",`
				`"secretsmanager:ListSecrets",`
				`"secretsmanager:TagResource"`

Link and unlink Git repositories	Yes	Yes	Yes	`"elasticmapreduce:LinkRepository",`
				`"elasticmapreduce:UnlinkRepository",`
				`"elasticmapreduce:ListRepositories",`
				`"elasticmapreduce:DescribeRepository"`

Action	Basic	Intermediate	Advanced	Associated actions

Create new clusters from predefined cluster templates	No	Yes	Yes	`"servicecatalog:SearchProducts",`
				`"servicecatalog:DescribeProduct",`
				`"servicecatalog:DescribeProductView",`
				`"servicecatalog:DescribeProvisioningParameters`
				`"servicecatalog:ProvisionProduct",`
				`"servicecatalog:UpdateProvisionedProduct",`
				`"servicecatalog:ListProvisioningArtifacts",`
				`"servicecatalog:DescribeRecord",`
				`"servicecatalog:ListLaunchPaths",`
				`"cloudformation:DescribeStackResources",`
				`"elasticmapreduce:ListClusters",`
				`"elasticmapreduce:DescribeCluster"`

Create new clusters by providing a cluster configuration	No	No	Yes	`"elasticmapreduce:RunJobFlow",`
				`"iam:PassRole",`
				`"elasticmapreduce:ListClusters",`
				`"elasticmapreduce:DescribeCluster"`

Assign a user to a Studio when you use IAM authentication mode. For more information, see Assign a user or group to an EMR Studio (p. 64).	No	No	No	`"elasticmapreduce:CreateStudioPresignedUrl"`

Describe network objects.	Yes	Yes	Yes	`{`
				`"Version": "2012-10-17",`
				`"Statement": [`
				`{`
				`"Sid":`
				`"DescribeNetwork",`
				`"Effect": "Allow",`
				`"Action": [`
				`"ec2:DescribeVpcs",`
				`"ec2:DescribeSubnets",`
				`"ec2:DescribeSecurityGroups"`
				`],`
				`"Resource": "*"`
				`}`
				`]`
				`}`

Action	Basic	Intermediate	Advanced	Associated actions
List IAM roles.	Yes	Yes	Yes	`{ "Version": "2012-10-17", "Statement": [ { "Sid": "ListIAMRoles", "Effect": "Allow", "Action": [ "iam:ListRoles" ], "Resource": "*" } ] }`

Example: Basic user policy

The following basic user policy allows most EMR Studio actions, but does not let a user create new Amazon EMR clusters.

> **Important**
> The example policy does not include the `CreateStudioPresignedUrl` permission, which you must allow for a user when you use IAM authentication mode. For more information, see Assign a user or group to an EMR Studio (p. 64).

The example policy includes `Condition` elements to enforce tag-based access control (TBAC) so that you can use the policy with the example service role for EMR Studio. For more information, see Create an EMR Studio service role (p. 42).

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowDefaultEC2SecurityGroupsCreationInVPCWithEMRTags",
         "Effect":"Allow",
         "Action":[
            "ec2:CreateSecurityGroup"
         ],
         "Resource":[
            "arn:aws:ec2:*:*:vpc/*"
         ],
         "Condition":{
            "StringEquals":{
               "aws:ResourceTag/for-use-with-amazon-emr-managed-policies":"true"
            }
         }
      },
      {
         "Sid":"AllowAddingEMRTagsDuringDefaultSecurityGroupCreation",
         "Effect":"Allow",
         "Action":[
            "ec2:CreateTags"
         ],
         "Resource":"arn:aws:ec2:*:*:security-group/*",
         "Condition":{
            "StringEquals":{
               "aws:RequestTag/for-use-with-amazon-emr-managed-policies":"true",
               "ec2:CreateAction":"CreateSecurityGroup"
```

```
            }
         }
      },
      {
         "Sid":"AllowSecretManagerListSecrets",
         "Action":[
            "secretsmanager:ListSecrets"
         ],
         "Resource":"*",
         "Effect":"Allow"
      },
      {
         "Sid":"AllowSecretCreationWithEMRTagsAndEMRStudioPrefix",
         "Effect":"Allow",
         "Action":"secretsmanager:CreateSecret",
         "Resource":"arn:aws:secretsmanager:*:*:secret:emr-studio-*",
         "Condition":{
            "StringEquals":{
               "aws:RequestTag/for-use-with-amazon-emr-managed-policies":"true"
            }
         }
      },
      {
         "Sid":"AllowAddingTagsOnSecretsWithEMRStudioPrefix",
         "Effect":"Allow",
         "Action":"secretsmanager:TagResource",
         "Resource":"arn:aws:secretsmanager:*:*:secret:emr-studio-*"
      },
      {
         "Sid":"AllowPassingServiceRoleForWorkspaceCreation",
         "Action":"iam:PassRole",
         "Resource":[
            "arn:aws:iam::*:role/<your-emr-studio-service-role>"
         ],
         "Effect":"Allow"
      },
      {
         "Sid":"AllowS3ListAndLocationPermissions",
         "Action":[
            "s3:ListAllMyBuckets",
            "s3:ListBucket",
            "s3:GetBucketLocation"
         ],
         "Resource":"arn:aws:s3:::*",
         "Effect":"Allow"
      },
      {
         "Sid":"AllowS3ReadOnlyAccessToLogs",
         "Action":[
            "s3:GetObject"
         ],
         "Resource":[
            "arn:aws:s3:::aws-logs-<aws-account-id>-<region>/elasticmapreduce/*"
         ],
         "Effect":"Allow"
      },
      {
         "Sid":"AllowConfigurationForWorkspaceCollaboration",
         "Action":[
            "elasticmapreduce:UpdateEditor",
            "elasticmapreduce:PutWorkspaceAccess",
            "elasticmapreduce:DeleteWorkspaceAccess",
            "elasticmapreduce:ListWorkspaceAccessIdentities"
         ],
         "Resource":"*",
         "Effect":"Allow",
```

```
            "Condition":{
               "StringEquals":{
                  "elasticmapreduce:ResourceTag/creatorUserId":"${aws:userId}"
               }
            }
         },
         {
            "Sid":"DescribeNetwork",
            "Effect":"Allow",
            "Action":[
               "ec2:DescribeVpcs",
               "ec2:DescribeSubnets",
               "ec2:DescribeSecurityGroups"
            ],
            "Resource":"*"
         },
         {
            "Sid":"ListIAMRoles",
            "Effect":"Allow",
            "Action":[
               "iam:ListRoles"
            ],
            "Resource":"*"
         }
      ]
}
```

Example: Intermediate user policy

The following intermediate user policy allows most EMR Studio actions, and lets a user create new Amazon EMR clusters using a cluster template.

> **Important**
> The example policy does not include the `CreateStudioPresignedUrl` permission, which you must allow for a user when you use IAM authentication mode. For more information, see Assign a user or group to an EMR Studio (p. 64).

The example policy includes `Condition` elements to enforce tag-based access control (TBAC) so that you can use the policy with the example service role for EMR Studio. For more information, see Create an EMR Studio service role (p. 42).

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowEMRBasicActions",
         "Action":[
            "elasticmapreduce:CreateEditor",
            "elasticmapreduce:DescribeEditor",
            "elasticmapreduce:ListEditors",
            "elasticmapreduce:StartEditor",
            "elasticmapreduce:StopEditor",
            "elasticmapreduce:DeleteEditor",
            "elasticmapreduce:OpenEditorInConsole",
            "elasticmapreduce:AttachEditor",
            "elasticmapreduce:DetachEditor",
            "elasticmapreduce:CreateRepository",
            "elasticmapreduce:DescribeRepository",
            "elasticmapreduce:DeleteRepository",
            "elasticmapreduce:ListRepositories",
            "elasticmapreduce:LinkRepository",
            "elasticmapreduce:UnlinkRepository",
            "elasticmapreduce:DescribeCluster",
            "elasticmapreduce:ListInstanceGroups",
```

```
            "elasticmapreduce:ListBootstrapActions",
            "elasticmapreduce:ListClusters",
            "elasticmapreduce:ListSteps",
            "elasticmapreduce:CreatePersistentAppUI",
            "elasticmapreduce:DescribePersistentAppUI",
            "elasticmapreduce:GetPersistentAppUIPresignedURL",
            "elasticmapreduce:GetOnClusterAppUIPresignedURL"
         ],
         "Resource":"*",
         "Effect":"Allow"
      },
      {
         "Sid":"AllowEMRContainersBasicActions",
         "Action":[
            "emr-containers:DescribeVirtualCluster",
            "emr-containers:ListVirtualClusters",
            "emr-containers:DescribeManagedEndpoint",
            "emr-containers:ListManagedEndpoints",
            "emr-containers:CreateAccessTokenForManagedEndpoint",
            "emr-containers:DescribeJobRun",
            "emr-containers:ListJobRuns"
         ],
         "Resource":"*",
         "Effect":"Allow"
      },
      {
         "Sid":"AllowSecretManagerListSecrets",
         "Action":[
            "secretsmanager:ListSecrets"
         ],
         "Resource":"*",
         "Effect":"Allow"
      },
      {
         "Sid":"AllowSecretCreationWithEMRTagsAndEMRStudioPrefix",
         "Effect":"Allow",
         "Action":"secretsmanager:CreateSecret",
         "Resource":"arn:aws:secretsmanager:*:*:secret:emr-studio-*",
         "Condition":{
            "StringEquals":{
               "aws:RequestTag/for-use-with-amazon-emr-managed-policies":"true"
            }
         }
      },
      {
         "Sid":"AllowAddingTagsOnSecretsWithEMRStudioPrefix",
         "Effect":"Allow",
         "Action":"secretsmanager:TagResource",
         "Resource":"arn:aws:secretsmanager:*:*:secret:emr-studio-*"
      },
      {
         "Sid":"AllowClusterTemplateRelatedIntermediateActions",
         "Action":[
            "servicecatalog:DescribeProduct",
            "servicecatalog:DescribeProductView",
            "servicecatalog:DescribeProvisioningParameters",
            "servicecatalog:ProvisionProduct",
            "servicecatalog:SearchProducts",
            "servicecatalog:UpdateProvisionedProduct",
            "servicecatalog:ListProvisioningArtifacts",
            "servicecatalog:ListLaunchPaths",
            "servicecatalog:DescribeRecord",
            "cloudformation:DescribeStackResources"
         ],
         "Resource":"*",
         "Effect":"Allow"
```

```
        },
        {
            "Sid":"AllowPassingServiceRoleForWorkspaceCreation",
            "Action":"iam:PassRole",
            "Resource":[
                "arn:aws:iam::*:role/<your-emr-studio-service-role>"
            ],
            "Effect":"Allow"
        },
        {
            "Sid":"AllowS3ListAndLocationPermissions",
            "Action":[
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource":"arn:aws:s3:::*",
            "Effect":"Allow"
        },
        {
            "Sid":"AllowS3ReadOnlyAccessToLogs",
            "Action":[
                "s3:GetObject"
            ],
            "Resource":[
                "arn:aws:s3:::aws-logs-<aws-account-id>-<region>/elasticmapreduce/*"
            ],
            "Effect":"Allow"
        },
        {
            "Sid":"AllowConfigurationForWorkspaceCollaboration",
            "Action":[
                "elasticmapreduce:UpdateEditor",
                "elasticmapreduce:PutWorkspaceAccess",
                "elasticmapreduce:DeleteWorkspaceAccess",
                "elasticmapreduce:ListWorkspaceAccessIdentities"
            ],
            "Resource":"*",
            "Effect":"Allow",
            "Condition":{
                "StringEquals":{
                    "elasticmapreduce:ResourceTag/creatorUserId":"${aws:userId}"
                }
            }
        },
        {
            "Sid":"DescribeNetwork",
            "Effect":"Allow",
            "Action":[
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups"
            ],
            "Resource":"*"
        },
        {
            "Sid":"ListIAMRoles",
            "Effect":"Allow",
            "Action":[
                "iam:ListRoles"
            ],
            "Resource":"*"
        }
    ]
}
```

Example: Advanced user policy

The following intermediate user policy allows all EMR Studio actions, and lets a user create new Amazon EMR clusters using a cluster template or by providing a cluster configuration.

> **Important**
> The example policy does not include the `CreateStudioPresignedUrl` permission, which you must allow for a user when you use IAM authentication mode. For more information, see Assign a user or group to an EMR Studio (p. 64).

The example policy includes `Condition` elements to enforce tag-based access control (TBAC) so that you can use the policy with the example service role for EMR Studio. For more information, see Create an EMR Studio service role (p. 42).

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowEMRBasicActions",
         "Action":[
            "elasticmapreduce:CreateEditor",
            "elasticmapreduce:DescribeEditor",
            "elasticmapreduce:ListEditors",
            "elasticmapreduce:StartEditor",
            "elasticmapreduce:StopEditor",
            "elasticmapreduce:DeleteEditor",
            "elasticmapreduce:OpenEditorInConsole",
            "elasticmapreduce:AttachEditor",
            "elasticmapreduce:DetachEditor",
            "elasticmapreduce:CreateRepository",
            "elasticmapreduce:DescribeRepository",
            "elasticmapreduce:DeleteRepository",
            "elasticmapreduce:ListRepositories",
            "elasticmapreduce:LinkRepository",
            "elasticmapreduce:UnlinkRepository",
            "elasticmapreduce:DescribeCluster",
            "elasticmapreduce:ListInstanceGroups",
            "elasticmapreduce:ListBootstrapActions",
            "elasticmapreduce:ListClusters",
            "elasticmapreduce:ListSteps",
            "elasticmapreduce:CreatePersistentAppUI",
            "elasticmapreduce:DescribePersistentAppUI",
            "elasticmapreduce:GetPersistentAppUIPresignedURL",
            "elasticmapreduce:GetOnClusterAppUIPresignedURL"
         ],
         "Resource":"*",
         "Effect":"Allow"
      },
      {
         "Sid":"AllowEMRContainersBasicActions",
         "Action":[
            "emr-containers:DescribeVirtualCluster",
            "emr-containers:ListVirtualClusters",
            "emr-containers:DescribeManagedEndpoint",
            "emr-containers:ListManagedEndpoints",
            "emr-containers:CreateAccessTokenForManagedEndpoint",
            "emr-containers:DescribeJobRun",
            "emr-containers:ListJobRuns"
         ],
         "Resource":"*",
         "Effect":"Allow"
      },
      {
         "Sid":"AllowSecretManagerListSecrets",
```

```
            "Action":[
               "secretsmanager:ListSecrets"
            ],
            "Resource":"*",
            "Effect":"Allow"
         },
         {
            "Sid":"AllowSecretCreationWithEMRTagsAndEMRStudioPrefix",
            "Effect":"Allow",
            "Action":"secretsmanager:CreateSecret",
            "Resource":"arn:aws:secretsmanager:*:*:secret:emr-studio-*",
            "Condition":{
               "StringEquals":{
                  "aws:RequestTag/for-use-with-amazon-emr-managed-policies":"true"
               }
            }
         },
         {
            "Sid":"AllowAddingTagsOnSecretsWithEMRStudioPrefix",
            "Effect":"Allow",
            "Action":"secretsmanager:TagResource",
            "Resource":"arn:aws:secretsmanager:*:*:secret:emr-studio-*"
         },
         {
            "Sid":"AllowClusterTemplateRelatedIntermediateActions",
            "Action":[
               "servicecatalog:DescribeProduct",
               "servicecatalog:DescribeProductView",
               "servicecatalog:DescribeProvisioningParameters",
               "servicecatalog:ProvisionProduct",
               "servicecatalog:SearchProducts",
               "servicecatalog:UpdateProvisionedProduct",
               "servicecatalog:ListProvisioningArtifacts",
               "servicecatalog:ListLaunchPaths",
               "servicecatalog:DescribeRecord",
               "cloudformation:DescribeStackResources"
            ],
            "Resource":"*",
            "Effect":"Allow"
         },
         {
            "Sid":"AllowEMRCreateClusterAdvancedActions",
            "Action":[
               "elasticmapreduce:RunJobFlow"
            ],
            "Resource":"*",
            "Effect":"Allow"
         },
         {
            "Sid":"AllowPassingServiceRoleForWorkspaceCreation",
            "Action":"iam:PassRole",
            "Resource":[
               "arn:aws:iam::*:role/<your-emr-studio-service-role>",
               "arn:aws:iam::*:role/EMR_DefaultRole",
               "arn:aws:iam::*:role/EMR_EC2_DefaultRole"
            ],
            "Effect":"Allow"
         },
         {
            "Sid":"AllowS3ListAndLocationPermissions",
            "Action":[
               "s3:ListAllMyBuckets",
               "s3:ListBucket",
               "s3:GetBucketLocation"
            ],
            "Resource":"arn:aws:s3:::*",
```

```
            "Effect":"Allow"
        },
        {
            "Sid":"AllowS3ReadOnlyAccessToLogs",
            "Action":[
                "s3:GetObject"
            ],
            "Resource":[
                "arn:aws:s3:::aws-logs-<aws-account-id>-<region>/elasticmapreduce/*"
            ],
            "Effect":"Allow"
        },
        {
            "Sid":"AllowConfigurationForWorkspaceCollaboration",
            "Action":[
                "elasticmapreduce:UpdateEditor",
                "elasticmapreduce:PutWorkspaceAccess",
                "elasticmapreduce:DeleteWorkspaceAccess",
                "elasticmapreduce:ListWorkspaceAccessIdentities"
            ],
            "Resource":"*",
            "Effect":"Allow",
            "Condition":{
                "StringEquals":{
                    "elasticmapreduce:ResourceTag/creatorUserId":"${aws:userId}"
                }
            }
        },
        {
            "Sid":"DescribeNetwork",
            "Effect":"Allow",
            "Action":[
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups"
            ],
            "Resource":"*"
        },
        {
            "Sid":"ListIAMRoles",
            "Effect":"Allow",
            "Action":[
                "iam:ListRoles"
            ],
            "Resource":"*"
        }
    ]
}
```

Create an EMR Studio

You can create an EMR Studio for your team using the Amazon EMR console or the AWS CLI. Creating a Studio instance is part of setting up Amazon EMR Studio.

Prerequisites

Before you create a Studio, make sure you've completed the previous tasks in Set up an Amazon EMR Studio (p. 38).

To create a Studio using the AWS CLI, you should have the latest version installed. For more information, see Installing or updating the latest version of the AWS CLI.

Important
Deactivate proxy management tools such as FoxyProxy or SwitchyOmega in the browser before you create a Studio. Active proxies can result in a **Network Failure** error message when you choose **Create Studio**.

Console

To create an EMR Studio using the EMR console

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/home
2. Choose **EMR Studio** from the left navigation.
3. Choose **Create Studio** to open the **Create a Studio** page.
4. Enter a **Studio name** and an optional **Description**.
5. If you use IAM authentication for the Studio, you can choose **Add new tag** to add one or more key-value tags of your choice to the Studio. You use tags to give specified users access to the Studio. For more information, see Assign a user or group to an EMR Studio (p. 64).

   You can also add tags to help you manage, identify, organize, and filter Studios. For more information, see Tagging AWS resources.
6. Under **Networking**, choose an Amazon Virtual Private Cloud (**VPC**) for the Studio from the dropdown list.
7. Under **Subnets**, select a maximum of five subnets in your VPC to associate with the Studio. You have the option to add more subnets after you create the Studio.
8. For **Security groups**, choose either the default security groups or custom security groups. For more information, see Define security groups to control EMR Studio network traffic (p. 73).

If you choose...	Do this...
The default EMR Studio security groups	To enable Git-based repository linking for the Studio, choose **Enable clusters/endpoints and Git repository**. Otherwise choose **Enable clusters/endpoints**.
Custom security groups for your Studio	• Under **Cluster/endpoint security group**, select the engine security group that you configured from the dropdown list. Your Studio uses this security group to allow inbound access from attached Workspaces. • Under **Workspace security group**, select the Workspace security group that you configured from the dropdown list. Your Studio uses this security group with Workspaces to provide outbound access to attached Amazon EMR clusters and publicly hosted Git repositories.

9. Under **Authentication**, choose an authentication mode for the Studio and provide information according to the following table. To learn more about authentication for EMR Studio, see Choose an authentication mode for Amazon EMR Studio (p. 38).

If you use...	Do this...
IAM authentication or federation	Choose a login method for the Studio.

If you use...	Do this...
	If you want federated users to log in using the Studio URL and credentials for your identity provider (IdP), select your IdP from the dropdown list, and enter your **Identity provider (IdP) login URL** and **RelayState** parameter name. For a list of IdP authentication URLs and RelayState names, see Identity provider RelayState parameters and authentication URLs (p. 63). Then, select your EMR Studio **Service role** from the dropdown list. For more information, see Create an EMR Studio service role (p. 42).
IAM Identity Center authentication	Select your EMR Studio **Service Role** and **User Role**. For more information, see Create an EMR Studio service role (p. 42) and Create an EMR Studio user role for IAM Identity Center authentication mode (p. 47).

10. Under **Workspace storage**, choose **Browse S3** to select your Amazon S3 bucket for backing up Workspaces and notebook files.

    Note
    Your EMR Studio service role must have read and write access to the bucket that you select.

11. Choose **Create Studio** to finish and navigate to the **Studios** page. Your new Studio appears in the list with details such as **Studio name**, **Creation date**, and **Studio access URL**.

After you create a Studio, follow the instructions in Assign a user or group to an EMR Studio (p. 64).

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 (^).

Example CLI command to create an EMR Studio with IAM authentication mode

The following example AWS CLI command creates an EMR Studio with IAM authentication mode. When you use IAM authentication or federation for the Studio, you don't specify a `--user-role`.

To let federated users log in using the Studio URL and credentials for your identity provider (IdP), specify your `--idp-auth-url` and `--idp-relay-state-parameter-name`. For a list of IdP authentication URLs and RelayState names, see Identity provider RelayState parameters and authentication URLs (p. 63).

```
aws emr create-studio \
--name <example-studio-name> \
--auth-mode IAM \
--vpc-id <example-vpc-id> \
--subnet-ids <subnet-id-1> <subnet-id-2>... <subnet-id-5>  \
--service-role <example-studio-service-role-name> \
--workspace-security-group-id <example-workspace-sg-id> \
```

```
--engine-security-group-id <example-engine-sg-id> \
--default-s3-location <example-s3-location> \
--idp-auth-url <https://EXAMPLE/login/> \
--idp-relay-state-parameter-name <example-RelayState>
```

Example CLI command to create an EMR Studio with IAM Identity Center authentication mode

The following AWS CLI example command creates an EMR Studio that uses IAM Identity Center authentication mode. When you use IAM Identity Center authentication, you must specify a `--user-role`.

For more information about IAM Identity Center authentication mode, see Set up IAM Identity Center authentication mode for Amazon EMR Studio (p. 40).

```
aws emr create-studio \
--name <example-studio-name> \
--auth-mode SSO \
--vpc-id <example-vpc-id> \
--subnet-ids <subnet-id-1> <subnet-id-2>... <subnet-id-5>  \
--service-role <example-studio-service-role-name> \
--user-role <example-studio-user-role-name> \
--workspace-security-group-id <example-workspace-sg-id> \
--engine-security-group-id <example-engine-sg-id> \
--default-s3-location <example-s3-location>
```

Example CLI output for `aws emr create-studio`

The following is an example of the output that appears after you create a Studio.

```
{
    StudioId: "es-123XXXXXXXXX",
    Url: "https://es-123XXXXXXXXX.emrstudio-prod.us-east-1.amazonaws.com"
}
```

For more information about the `create-studio` command, see *AWS CLI Command Reference*.

Identity provider RelayState parameters and authentication URLs

When you use IAM federation, and you want users to log in using your Studio URL and credentials for your identity provider (IdP), you can specify your **Identity provider (IdP) login URL** and **RelayState** parameter name when you Create an EMR Studio (p. 60).

The following table shows the standard authentication URL and RelayState parameter name for some popular identity providers.

Identity provider	Parameter	Authentication URL
Auth0	`RelayState`	`https://<sub_domain>.auth0.com/samlp/<app_id>`
Google accounts	`RelayState`	`https://accounts.google.com/o/saml2/initsso?idpid=<idp_id>&spid=<sp_id>&forceauthn=false`
Microsoft Azure	`RelayState`	`https://myapps.microsoft.com/signin/<app_name>/<app_id>?tenantId=<tenant_id>`

Identity provider	Parameter	Authentication URL
Okta	`RelayState`	`https://<sub_domain>.okta.com/app/<app_name>/<app_id>/sso/saml`
PingFederate	`TargetResource`	`https://<host>/idp/<idp_id>/startSSO.ping?PartnerSpId=<sp_id>`
PingOne	`TargetResource`	`https://sso.connect.pingidentity.com/sso/sp/initsso?saasid=<app_id>&idpid=<idp_id>`

Assign and manage EMR Studio users

After you create an EMR Studio, you can assign users and groups to it. The method you use to assign, update, and remove users depends on the Studio authentication mode.

- When you use IAM authentication mode, you configure EMR Studio user assignment and permissions in IAM or with IAM and your identity provider.
- With IAM Identity Center authentication mode, you use the Amazon EMR management console or the AWS CLI to manage users.

To learn more about authentication for Amazon EMR Studio, see Choose an authentication mode for Amazon EMR Studio (p. 38).

Assign a user or group to an EMR Studio

IAM

When you use Set up IAM authentication mode for Amazon EMR Studio (p. 39), you must allow the `CreateStudioPresignedUrl` action in a user's IAM permissions policy and restrict the user to a particular Studio. You can include `CreateStudioPresignedUrl` in your User permissions for IAM authentication mode (p. 29) or use a separate policy.

To restrict a user to a Studio (or set of Studios), you can use attribute-based access control (ABAC) or specify the Amazon Resource Name (ARN) of a Studio in the `Resource` element of the permissions policy.

Example Assign a user to a Studio using a Studio ARN

The following example policy gives a user access to a particular EMR Studio by allowing the `CreateStudioPresignedUrl` action and specifying the Studio's Amazon Resource Name (ARN) in the `Resource` element.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCreateStudioPresignedUrl",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:CreateStudioPresignedUrl"
            ],
            "Resource": "arn:aws:elasticmapreduce:<region>:<account-id>:studio/<studio-
id>"
        }
    ]
```

```
}
```

Example Assign a user to a Studio with ABAC for IAM authentication

There are multiple ways to configure attribute-based access control (ABAC) for a Studio. For example, you might attach one or more tags to an EMR Studio, and then create an IAM policy that restricts the `CreateStudioPresignedUrl` action to a particular Studio or set of Studios with those tags.

You can add tags during or after Studio creation. To add tags to an existing Studio, you can use the AWS CLI`emr add-tags` command. The following example adds a tag with the key-value pair `Team = Data Analytics` to an EMR Studio.

```
aws emr add-tags --resource-id <example-studio-id> --tags team="Data Analytics"
```

The following example permissions policy allows the `CreateStudioPresignedUrl` action for EMR Studios with the tag key-value pair `Team = DataAnalytics`. For more information about using tags to control access, see Controlling access to and for IAM users and roles using tags or Controlling access to AWS resources using tags.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCreateStudioPresignedUrl",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:CreateStudioPresignedUrl"
            ],
            "Resource": "arn:aws:elasticmapreduce:<region>:<account-id>:studio/*",
            "Condition": {
                "StringEquals": {
                    "elasticmapreduce:ResourceTag/Team": "Data Analytics"
                }
            }
        }
    ]
}
```

Example Assign a user to a Studio using the aws:SourceIdentity global condition key

When you use IAM federation, you can use the global condition key `aws:SourceIdentity` in a permissions policy to give users Studio access when they assume your IAM role for federation.

You must first configure your identity provider (IdP) to return an identifying string, such as an email address or username, when a user authenticates and assumes your IAM role for federation. IAM sets the global condition key `aws:SourceIdentity` to the identifying string returned by your IdP.

For more information, see the How to relate IAM role activity to corporate identity blog post in the AWS Security Blog and the aws:SourceIdentity entry in the global condition keys reference.

The following example policy allows the `CreateStudioPresignedUrl` action and gives users with an `aws:SourceIdentity` that matches the *`<example-source-identity>`* access to the EMR Studio specified by *`<example-studio-arn>`*.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Action": "elasticmapreduce:CreateStudioPresignedUrl",
      "Resource": "<example-studio-arn>",
      "Condition": {
        "StringLike": {
          "aws:SourceIdentity": "<example-source-identity>"
        }
      }
    }
  ]
}
```

IAM Identity Center

When you assign a user or group to an EMR Studio, you specify a session policy that defines fine-grained permissions, such as the ability to create a new EMR cluster, for that user or group. Amazon EMR stores these session policy mappings. You can update a user or group's session policy after assignment.

> **Note**
> The final permissions for a user or group is an intersection of the permissions defined in your EMR Studio user role and the permissions defined in the session policy for that user or group. If a user belongs to more than one group assigned to the Studio, EMR Studio uses a union of permissions for that user.

To assign users or groups to an EMR Studio using the EMR console

1. Sign in to the AWS Management Console and open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose **EMR Studio** from the left navigation.
3. Choose your Studio name from the **Studios** list, or select the Studio and choose **View details**, to open the Studio detail page.
4. Choose **Add Users** to see the **Users** and **Groups** search table.
5. Select the **Users** tab or the **Groups** tab, and enter a search term in the search bar to find a user or group.
6. Select one or more users or groups from the search results list. You can switch back and forth between the **Users** tab and the **Groups** tab.
7. After you select users and groups to add to the Studio, choose **Add**. You should see the users and groups appear in the **Studio users** list. It might take a few seconds for the list to refresh.
8. Follow the instructions in Update permissions for a user or group assigned to a Studio (p. 67) to refine the Studio permissions for a user or group.

To assign a user or group to an EMR Studio using the AWS CLI

Insert your own values for the following `create-studio-session-mapping` arguments. For more information about the `create-studio-session-mapping` command, see the *AWS CLI Command Reference*.

- **`--studio-id`** – The ID of the Studio you want to assign the user or group to. For instructions on how to retrieve a Studio ID, see View Studio details (p. 69).
- **`--identity-name`** – The name of the user or group from the Identity Store. For more information, see UserName for users and DisplayName for groups in the *Identity Store API Reference*.
- **`--identity-type`** – Use either USER or GROUP to specify the identity type.
- **`--session-policy-arn`** – The Amazon Resource Name (ARN) for the session policy you want to associate with the user or group. For example, **`arn:aws:iam::<aws-account-`**

id>*:policy/*EMRStudio_Advanced_User_Policy. For more information, see Create permissions policies for EMR Studio users (p. 48).

```
aws emr create-studio-session-mapping \
 --studio-id <example-studio-id> \
 --identity-name <example-identity-name> \
 --identity-type <USER-or-GROUP> \
 --session-policy-arn <example-session-policy-arn>
```

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 (^).

Use the `get-studio-session-mapping` command to verify the new assignment. Replace *`<example-identity-name>`* with the IAM Identity Center name of the user or group that you updated.

```
aws emr get-studio-session-mapping \
 --studio-id <example-studio-id> \
 --identity-type <USER-or-GROUP> \
 --identity-name <user-or-group-name> \
```

Update permissions for a user or group assigned to a Studio

IAM

To update user or group permissions when you use IAM authentication mode, use IAM to change the IAM permissions policies attached to your IAM identities (users, groups, or roles).

For more information, see User permissions for IAM authentication mode (p. 29).

IAM Identity Center

To update EMR Studio permissions for a user or group using the console

1. Sign in to the AWS Management Console and open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose **EMR Studio** from the left navigation.
3. Choose your Studio name from the **Studios** list, or select the Studio and choose **View details**, to open the Studio detail page.
4. In the **Studio users** list on the Studio detail page, search for the user or group you want to update. You can search by name or identity type.
5. Select the user or group that you want to update and choose **Assign policy** to open the **Session policy** dialog box.
6. Select a policy to apply to the user or group that you chose in step 5, and choose **Apply policy**. The **Studio users** list should display the policy name in the **Session policy** column for the user or group that you updated.

To update EMR Studio permissions for a user or group using the AWS CLI

Insert your own values for the following `update-studio-session-mappings` arguments. For more information about the `update-studio-session-mappings` command, see the *AWS CLI Command Reference*.

```
aws emr update-studio-session-mapping \
```

```
 --studio-id <example-studio-id> \
 --identity-name <name-of-user-or-group-to-update> \
 --session-policy-arn <new-session-policy-arn-to-apply> \
 --identity-type <USER-or-GROUP> \
```

Use the `get-studio-session-mapping` command to verify the new session policy assignment. Replace *`<example-identity-name>`* with the IAM Identity Center name of the user or group that you updated.

```
aws emr get-studio-session-mapping \
 --studio-id <example-studio-id> \
 --identity-type <USER-or-GROUP> \
 --identity-name <user-or-group-name> \
```

Remove a user or group from a Studio

IAM

To remove a user or group from an EMR Studio when you use IAM authentication mode, you must revoke the user's access to the Studio by reconfiguring the user's IAM permissions policy.

In the following example policy, assume that you have an EMR Studio with the tag key-value pair `Team = Quality Assurance`. According to the policy, the user can access Studios tagged with the `Team` key whose value is equal to either `Data Analytics` or `Quality Assurance`. To remove the user from the Studio tagged with `Team = Quality Assurance`, remove `Quality Assurance` from the list of tag values.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCreateStudioPresignedUrl",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:CreateStudioPresignedUrl"
            ],
            "Resource": "arn:aws:elasticmapreduce:<region>:<account-id>:studio/*",
            "Condition": {
                "StringEquals": {
                    "emr:ResourceTag/Team": [
                        "Data Analytics",
                        "Quality Assurance"
                  ]
                }
            }
        }
    ]
}
```

IAM Identity Center

To remove a user or group from an EMR Studio using the console

1. Sign in to the AWS Management Console and open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose **EMR Studio** from the left navigation.
3. Choose your Studio name from the **Studios** list, or select the Studio and choose **View details**, to open the Studio detail page.

4. In the **Studio users** list on the Studio detail page, find the user or group you want to remove from the Studio. You can search by name or identity type.
5. Select the user or group that you want to delete, choose **Delete** and confirm. The user or group that you deleted disappears from the **Studio users** list.

To remove a user or group from an EMR Studio using the AWS CLI

Insert your own values for the following `delete-studio-session-mapping` arguments. For more information about the `delete-studio-session-mapping` command, see the *AWS CLI Command Reference*.

```
aws emr delete-studio-session-mapping \
 --studio-id <example-studio-id> \
 --identity-type <USER-or-GROUP> \
 --identity-name <name-of-user-or-group-to-delete> \
```

Manage an Amazon EMR Studio

This section includes instructions to help you monitor, update, or delete an EMR Studio resource. For information about assigning users or updating user permissions, see Assign and manage EMR Studio users (p. 64).

View Studio details

Console

To view details about an EMR Studio using the console

1. Sign in to the AWS Management Console and open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose **EMR Studio** from the left navigation.
3. Select the Studio from the **Studios** list to open the Studio detail page. The Studio detail page includes **Studio setting** information, such as the Studio **Description**, **VPC**, and **Subnets**.

CLI

To retrieve details for an EMR Studio by Studio ID using the AWS CLI

Use the following `describe-studio` AWS CLI command to fetch detailed information about a particular EMR Studio. For more information, see the *AWS CLI Command Reference*.

```
aws emr describe-studio \
 --studio-id <id-of-studio-to-describe> \
```

To retrieve a list of EMR Studios using the AWS CLI

Use the following `list-studios` AWS CLI command. For more information, see the *AWS CLI Command Reference*.

```
aws emr list-studios
```

The following is an example return value for the `list-studios` command in JSON format.

```
{
    "Studios": [
        {
            "AuthMode": "IAM",
            "VpcId": "vpc-b21XXXXX",
            "Name": "example-studio-name",
            "Url": "https://es-7HWP74SNGDXXXXXXXXXXXXXX.emrstudio-prod.us-
east-1.amazonaws.com",
            "CreationTime": 1605672582.781,
            "StudioId": "es-7HWP74SNGDXXXXXXXXXXXXXX",
            "Description": "example studio description"
        }
    ]
}
```

Monitor Amazon EMR Studio actions

View EMR Studio and API activity

EMR Studio is integrated with AWS CloudTrail, a service that provides a record of actions taken by an IAM user, by an IAM role, or by another AWS service in EMR Studio. CloudTrail captures API calls for EMR Studio as events. You can view events using the CloudTrail console at https://console.aws.amazon.com/cloudtrail/.

EMR Studio events provide information such as which Studio or IAM user makes a request, and what kind of request it is.

> **Note**
> On-cluster actions such as running notebook jobs do not emit AWS CloudTrail.

You can also create a trail for continuous delivery of EMR Studio CloudTrail events to an Amazon S3 bucket. For more information, see the *AWS CloudTrail User Guide*.

Example CloudTrail Event: An IAM User Calls the DescribeStudio API

The following is an example AWS CloudTrail event that is created when an IAM user, `admin`, calls the DescribeStudio API. CloudTrail records the user name as `admin`.

> **Note**
> To protect Studio details, the EMR Studio API event for DescribeStudio excludes a value for `responseElements`.

```
{
   "eventVersion":"1.08",
   "userIdentity":{
      "type":"IAMUser",
      "principalId":"AIDXXXXXXXXXXXXXXXXXX",
      "arn":"arn:aws:iam::653XXXXXXXXX:user/admin",
      "accountId":"653XXXXXXXXX",
      "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
      "userName":"admin"
   },
   "eventTime":"2021-01-07T19:13:58Z",
   "eventSource":"elasticmapreduce.amazonaws.com",
   "eventName":"DescribeStudio",
   "awsRegion":"us-east-1",
   "sourceIPAddress":"72.XX.XXX.XX",
   "userAgent":"aws-cli/1.18.188 Python/3.8.5 Darwin/18.7.0 botocore/1.19.28",
   "requestParameters":{
```

```
      "studioId":"es-9O5XXXXXXXXXXXXXXXXXXXXX"
   },
   "responseElements":null,
   "requestID":"0fxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
   "eventID":"b0xxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
   "readOnly":true,
   "eventType":"AwsApiCall",
   "managementEvent":true,
   "eventCategory":"Management",
   "recipientAccountId":"653XXXXXXXXX"
}
```

View Spark user and job activity

To view Spark job activity by Amazon EMR Studio users, you can configure user impersonation on a cluster. With user impersonation, each Spark job that is submitted from a Workspace is associated with the Studio user who ran the code.

When user impersonation is enabled, Amazon EMR creates an HDFS user directory on the cluster's master node for each user that runs code in the Workspace. For example, if user `studio-user-1@example.com` runs code, you can connect to the master node and see that `hadoop fs -ls /user` has a directory for `studio-user-1@example.com`.

To set up Spark user impersonation, set the following properties in the following configuration classifications:

- `core-site`
- `livy-conf`

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

To view history server pages, see Debug applications and jobs with EMR Studio (p. 94). You can also connect to the master node of the cluster using SSH to view application web interfaces. For more information, see View web interfaces hosted on Amazon EMR clusters (p. 599).

Update an Amazon EMR Studio

After you create an EMR Studio, you can update the following attributes using the AWS CLI:

- Name
- Description
- Default S3 location
- Subnets

To update an EMR Studio using the AWS CLI

Use the `update-studio` AWS CLI command to update an EMR Studio. For more information, see the *AWS CLI Command Reference*.

> **Note**
> You can associated a Studio with a maximum of 5 subnets. These subnets must belong to the same VPC as the Studio. The list of subnet IDs that you submit to the `update-studio` command can include new subnet IDs, but must also include all of the subnet IDs that you already associated with the Studio. You can't remove subnets from a Studio.

```
aws emr update-studio \
 --studio-id <example-studio-id-to-update> \
 --name <example-new-studio-name> \
 --subnet-ids <old-subnet-id-1 old-subnet-id-2 old-subnet-id-3 new-subnet-id> \
```

To verify the changes, use the `describe-studio` AWS CLI command and specify your Studio ID. For more information, see the *AWS CLI Command Reference*.

```
aws emr describe-studio \
 --studio-id <id-of-updated-studio> \
```

Delete an Amazon EMR Studio and Workspaces

When you delete a Studio, EMR Studio deletes all of the IAM Identity Center user and group assignments that are associated with the Studio.

> **Note**
> When you delete a Studio, Amazon EMR does *not* delete the Workspaces associated with that Studio. You must delete the Workspaces in your Studio separately.

Delete Workspaces

Console

Since each EMR Studio Workspace is an EMR notebook instance, you can use the Amazon EMR management console to delete Workspaces. You can delete Workspaces using the Amazon EMR console before or after you delete your Studio

To delete a Workspace using the Amazon EMR console

1. Sign in to the AWS Management Console and open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose **Notebooks**.
3. Select the Workspace(s) that you want to delete.
4. Choose **Delete**, then choose **Delete** again to confirm.
5. Follow the instructions for Deleting objects in the *Amazon Simple Storage Service Console User Guide* to remove the notebook files associated with the deleted Workspace from Amazon S3.

EMR Studio UI

To delete a Workspace and its associated backup files from EMR Studio

1. Log in to your EMR Studio with your Studio access URL and choose **Workspaces** from the left navigation.

2. Find your Workspace in the list, then select the check box next to its name. You can select multiple Workspaces to delete at the same time.
3. Choose **Delete** in the upper right of the **Workspaces** list and confirm that you want to delete the selected Workspaces. Choose **Delete** to confirm.
4. Follow the instructions for Deleting objects in the *Amazon Simple Storage Service Console User Guide* to remove the notebook files associated with the deleted Workspace from Amazon S3. If you did not create the Studio, consult your Studio administrator to determine the Amazon S3 backup location for the deleted Workspace.

Delete an EMR Studio

Console

To delete an EMR Studio using the console

1. Sign in to the AWS Management Console and open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose **EMR Studio** from the left navigation.
3. Select the Studio from the **Studios** list and choose **Delete**.

CLI

To delete an EMR Studio using the AWS CLI

Use the `delete-studio` AWS CLI command to delete an EMR Studio. For more information, see the *AWS CLI Command Reference*.

```
aws emr delete-studio --studio-id <id-of-studio-to-delete>
```

Define security groups to control EMR Studio network traffic

About the EMR Studio security groups

Amazon EMR Studio uses two security groups to control network traffic between Workspaces in the Studio and an attached Amazon EMR cluster running on Amazon EC2:

- An **engine security group** that uses port 18888 to communicate with an attached Amazon EMR cluster running on Amazon EC2.
- A **Workspace security group** associated with the Workspaces in a Studio. This security group includes an outbound HTTPS rule to allow the Workspace to route traffic to the internet and must allow outbound traffic to the internet on port 443 to enable linking Git repositories to a Workspace.

EMR Studio uses these security groups in addition to any security groups associated with an EMR cluster attached to a Workspace.

You must create these security groups when you use the AWS CLI to create a Studio.

Note
You can customize the security groups for EMR Studio with rules tailored to your environment, but you must include the rules noted on this page. Your Workspace security group can't

allow any inbound traffic, and the engine security group must allow inbound traffic from the Workspace security group.

Use the Default EMR Studio Security Groups

When you use the EMR console, you can choose the following default security groups. The default security groups are created by EMR Studio on your behalf, and include the minimum required inbound and outbound rules for Workspaces in an EMR Studio.

- `DefaultEngineSecurityGroup`
- `DefaultWorkspaceSecurityGroupGit` or `DefaultWorkspaceSecurityGroupWithoutGit`

Prerequisites

To create the security groups for EMR Studio, you need an Amazon Virtual Private Cloud (VPC) for the Studio. You choose this VPC when you create the security groups. This should be the same VPC that you specify when you create the Studio. If you plan to use Amazon EMR on EKS with EMR Studio, choose the VPC for your Amazon EKS cluster worker nodes.

Instructions

Follow the instructions in Creating a security group in the *Amazon EC2 User Guide for Linux Instances* to create an engine security group and a Workspace security group in your VPC. The security groups must include the rules summarized in the following tables.

When you create security groups for EMR Studio, note the IDs for both. You specify each security group by ID when you create a Studio.

Engine security group

EMR Studio uses port 18888 to communicate with an attached cluster.

Inbound rules

Type	Protocol	Port	Destination	Description
TCP	TCP	18888	Your EMR Studio Workspace security group.	Allow traffic from any resources in the Workspace security group for EMR Studio.

Workspace security group

This security group is associated with the Workspaces in an EMR Studio.

Outbound rules

Type	Protocol	Port	Destination	Description
TCP	TCP	18888	Your EMR Studio engine security group.	Allow traffic to any resources in the Engine security group for EMR Studio.
HTTPS	TCP	443	0.0.0.0/0	Allow traffic to the internet to link publicly

Type	Protocol	Port	Destination	Description
				hosted Git repositories to Workspaces.

Create AWS CloudFormation templates for Amazon EMR Studio

About EMR Studio cluster templates

You can create AWS CloudFormation templates to help EMR Studio users launch new Amazon EMR clusters in a Workspace. CloudFormation templates are formatted text files in JSON or YAML. In a template, you describe a stack of AWS resources and tell CloudFormation how to provision those resources for you. For EMR Studio, you can create one or more templates that describe an Amazon EMR cluster.

You organize your templates in AWS Service Catalog. AWS Service Catalog lets you create and manage commonly deployed IT services called *products* on AWS. You collect your templates as products in a *portfolio* that you share with your EMR Studio users. After you create cluster templates, Studio users can launch a new cluster for a Workspace with one of your templates. Users must have permission to create new clusters from templates. You can set user permissions in your EMR Studio permissions policies (p. 47).

To learn more about CloudFormation templates, see Templates in the *AWS CloudFormation User Guide*. For more information about AWS Service Catalog, see What is AWS Service Catalog.

The following video demonstrates how to set up cluster templates in AWS Service Catalog for EMR Studio. You can also learn more in the Build a self-service environment for each line of business using Amazon EMR and AWS Service Catalog blog post.

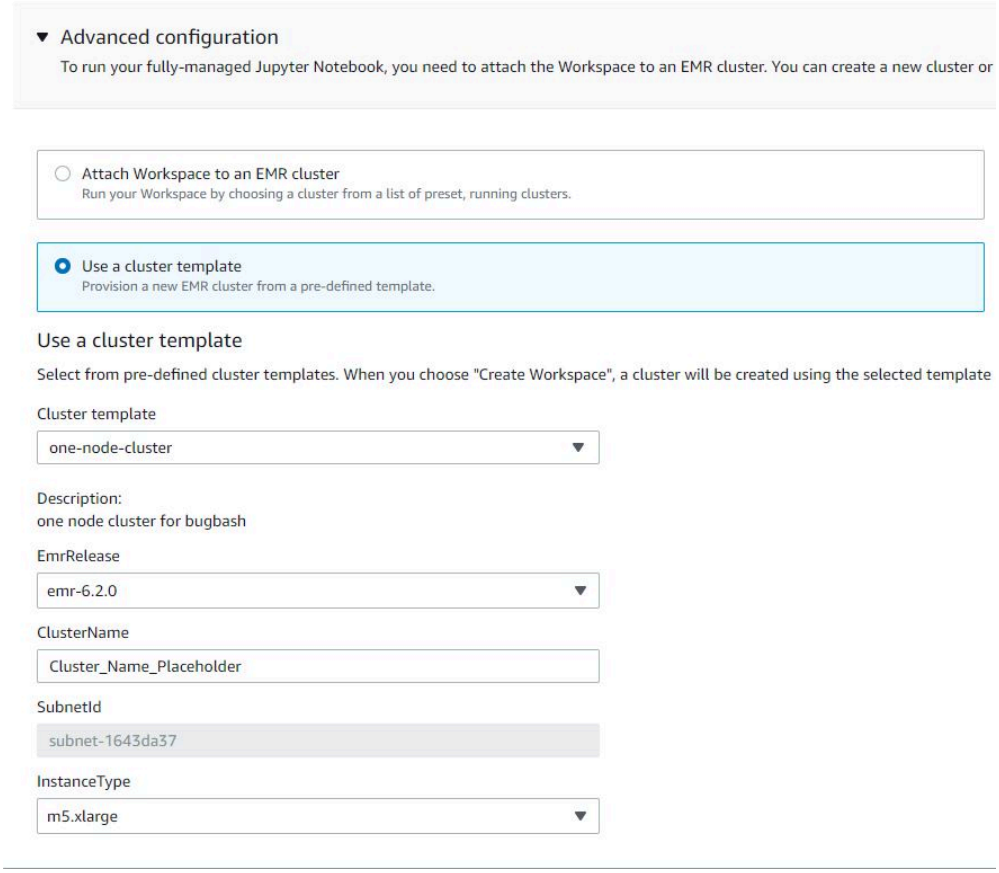
Optional template parameters

You can include additional options in the `Parameters` section of your template. *Parameters* let Studio users input or select custom values for a cluster. For example, you could add a parameter that lets users select a particular Amazon EMR release. For more information, see Parameters in the *AWS CloudFormation User Guide*.

The following example `Parameters` section defines additional input parameters such as `ClusterName`, `EmrRelease` version, and `ClusterInstanceType`.

```
Parameters:
  ClusterName:
    Type: "String"
    Default: "Cluster_Name_Placeholder"
  EmrRelease:
    Type: "String"
    Default: "emr-6.2.0"
    AllowedValues:
    - "emr-6.2.0"
    - "emr-5.32.0"
  ClusterInstanceType:
    Type: "String"
    Default: "m5.xlarge"
    AllowedValues:
    - "m5.xlarge"
    - "m5.2xlarge"
```

When you add parameters, Studio users see additional form options after selecting a cluster template. The following image shows additional form options for **EmrRelease** version, **ClusterName**, and **InstanceType**.



Prerequisites

Before you create a cluster template, make sure you have IAM permissions to access the AWS Service Catalog administrator console view. You also need the required IAM permissions to perform AWS Service Catalog administrative tasks. For more information, see Grant permissions to AWS Service Catalog administrators.

Instructions

To create EMR cluster templates using AWS Service Catalog

1. Create one or more CloudFormation templates. Where you store your templates is up to you. Since templates are formatted text files, you can upload them to Amazon S3 or keep them in your local file system. To learn more about CloudFormation templates, see Templates in the *AWS CloudFormation User Guide*.

   Use the following rules to name your templates, or check your names against the pattern `[a-zA-Z0-9][a-zA-Z0-9._-]*`.

   - Template names must start with a letter or a number.
   - Template names can only consist of letters, numbers, periods (.), underscores (_), and hyphens (-).

Each cluster template that you create must include the following options:

Input parameters

- ClusterName – A name for the cluster to help users identify it after it has been provisioned.

Output

- ClusterId – The ID of the newly-provisioned EMR cluster.

Following is an example AWS CloudFormation template in YAML format for a cluster with two nodes. The example template includes the required template options and defines additional input parameters for `EmrRelease` and `ClusterInstanceType`.

```
AWSTemplateFormatVersion: 2010-09-09

Parameters:
  ClusterName:
    Type: "String"
    Default: "Example_Two_Node_Cluster"
  EmrRelease:
    Type: "String"
    Default: "emr-6.2.0"
    AllowedValues:
    - "emr-6.2.0"
    - "emr-5.32.0"
  ClusterInstanceType:
    Type: "String"
    Default: "m5.xlarge"
    AllowedValues:
    - "m5.xlarge"
    - "m5.2xlarge"

Resources:
  EmrCluster:
    Type: AWS::EMR::Cluster
    Properties:
      Applications:
      - Name: Spark
      - Name: Livy
      - Name: JupyterEnterpriseGateway
      - Name: Hive
      EbsRootVolumeSize: '10'
      Name: !Ref ClusterName
      JobFlowRole: EMR_EC2_DefaultRole
      ServiceRole: EMR_DefaultRole
      ReleaseLabel: !Ref EmrRelease
      VisibleToAllUsers: true
      LogUri:
        Fn::Sub: 's3://aws-logs-${AWS::AccountId}-${AWS::Region}/elasticmapreduce/'
      Instances:
        TerminationProtected: false
        Ec2SubnetId: 'subnet-ab12345c'
        MasterInstanceGroup:
          InstanceCount: 1
          InstanceType: !Ref ClusterInstanceType
        CoreInstanceGroup:
          InstanceCount: 1
          InstanceType: !Ref ClusterInstanceType
          Market: ON_DEMAND
```

```
          Name: Core

Outputs:
  ClusterId:
    Value:
      Ref: EmrCluster
    Description: The ID of the EMR Cluster
```

2. Create a portfolio for your cluster templates in the same AWS account as your Studio.
   a. Open the AWS Service Catalog console at https://console.aws.amazon.com/servicecatalog/.
   b. Choose **Portfolios** in the left navigation menu.
   c. Enter the requested information on the **Create portfolio** page.
   d. Choose **Create**. AWS Service Catalog creates the portfolio and displays the portfolio details.
3. Use the following steps to add your cluster templates as AWS Service Catalog products.
   a. Navigate to the **Products** page under **Administration** in the AWS Service Catalog management console.
   b. Choose **Upload new product**.
   c. Enter a **Product name** and **Owner**.
   d. Specify your template file under **Version details**.
   e. Choose **Review** to review your product settings, then choose **Create product**.
4. Complete the following steps to add your products to your portfolio.
   a. Navigate to the **Products** page in the AWS Service Catalog management console.
   b. Choose your product, choose **Actions**, then choose **Add product to portfolio**.
   c. Choose your portfolio, then choose **Add product to portfolio**.
5. Create a launch constraint for your products. A launch constraint is an IAM role that specifies user permissions for launching a product. You can tailor your launch constraints, but must allow permissions to use CloudFormation, Amazon EMR, and AWS Service Catalog. For more information and instructions, see AWS Service Catalog launch constraints.
6. Apply your launch constraint to each product in your portfolio. You must apply the launch constraint to each product individually.
   a. Select your portfolio from the **Portfolios** page in the AWS Service Catalog management console.
   b. Choose the **Constraints** tab and choose **Create constraint**.
   c. Choose your product and choose **Launch** under **Constraint type**. Choose **Continue**.
   d. Select your launch constraint role in the **Launch constraint** section, then choose **Create**.
7. Grant access to your portfolio.
   a. Select your portfolio from the **Portfolios** page in the AWS Service Catalog management console.
   b. Expand the **Groups, roles, and users** tab and choose **Add groups, roles, users**.
   c. Search for your EMR Studio IAM role in the **Roles** tab, select your role, and choose **Add access**.

If you use....	Grant access to...
IAM authentication	Your native IAM users
IAM federation	Your IAM role for federation
IAM Identity Center federation	Your EMR Studio user role (p. 47)

Establish access and permissions for Git-based repositories

EMR Studio supports the following Git-based services:

- AWS CodeCommit
- GitHub
- Bitbucket
- GitLab

To let EMR Studio users associate a Git repository with a Workspace, set up the following access and permissions requirements. You can also configure Git-based repositories that you host in a private network by following the instructions in Configure a privately hosted Git repository for EMR Studio (p. 79).

Cluster internet access

Both Amazon EMR clusters running on Amazon EC2 and Amazon EMR on EKS clusters attached to Studio Workspaces must be in a private subnet that uses a network address translation (NAT) gateway, or they must be able to access the internet through a virtual private gateway. For more information, see Amazon VPC options (p. 307).

The security groups that you use with EMR Studio must also include an outbound rule that allows Workspaces to route traffic to the internet from an attached EMR cluster. For more information, see Define security groups to control EMR Studio network traffic (p. 73).

> **Important**
> If the network interface is in a public subnet, it won't be able to communicate with the internet through an internet gateway (IGW).

Permissions for AWS Secrets Manager

To let EMR Studio users access Git repositories with secrets stored in AWS Secrets Manager, add a permissions policy to the service role for EMR Studio (p. 42) that allows the `secretsmanager:GetSecretValue` operation.

For information about how to link Git-based repositories to Workspaces, see Link Git-based repositories to an EMR Studio Workspace (p. 92).

Configure a privately hosted Git repository for EMR Studio

Use the following instructions to configure privately hosted repositories for Amazon EMR Studio. Provide a configuration file with information about your DNS and Git servers. EMR Studio uses this information to configure Workspaces that can route traffic to your self-managed repositories.

> **Note**
> If you configure `DnsServerIpV4`, EMR Studio uses your DNS server to resolve both your `GitServerDnsName` and your Amazon EMR endpoint, such as `elasticmapreduce.us-east-1.amazonaws.com`. To set up an endpoint for Amazon EMR, connect to your endpoint through the VPC that you're using with your Studio. This ensures that the Amazon EMR endpoint resolves to a private IP. For more information, see Connect to Amazon EMR using an interface VPC endpoint (p. 546).

Prerequisites

Before you configure a privately hosted Git repository for EMR Studio, you need an Amazon S3 storage location where EMR Studio can back up the Workspaces and notebook files in the Studio. Use the same S3 bucket that you specify when you create a Studio.

To configure one or more privately hosted Git repositories for EMR Studio

1. Create a configuration file using the following template. Include the following values for each Git server that you want to specify in your configuration:

   - **`DnsServerIpV4`** - The IPv4 address of your DNS server. If you provide values for both `DnsServerIpV4` and `GitServerIpV4List`, the value for `DnsServerIpV4` takes precedence and EMR Studio uses `DnsServerIpV4` to resolve your `GitServerDnsName`.

     Note
     To use privately hosted Git repositories, your DNS server must allow inbound access from EMR Studio. We urge you to secure your DNS server against other, unauthorized access.

   - **`GitServerDnsName`** - The DNS name of your Git server. For example `"git.example.com"`.
   - **`GitServerIpV4List`** - A list of IPv4 addresses that belong to your Git servers.

```
[
    {
        "Type": "PrivatelyHostedGitConfig",
        "Value": [
            {
                "DnsServerIpV4": "<10.24.34.xxx>",
                "GitServerDnsName": "<enterprise.git.com>",
                "GitServerIpV4List": [
                    "<xxx.xxx.xxx.xxx>",
                    "<xxx.xxx.xxx.xxx>"
                ]
            },
            {
                "DnsServerIpV4": "<10.24.34.xxx>",
                "GitServerDnsName": "<git.example.com>",
                "GitServerIpV4List": [
                    "<xxx.xxx.xxx.xxx>",
                    "<xxx.xxx.xxx.xxx>"
                ]
            }
        ]
    }
]
```

2. Save your configuration file as `configuration.json`.
3. Upload the configuration file into your Amazon S3 storage location in a folder called `life-cycle-configuration`. For example, if your default S3 location is `s3://DOC-EXAMPLE-BUCKET/studios`, your configuration file would be in `s3://DOC-EXAMPLE-BUCKET/studios/`**`life-cycle-configuration/configuration.json`**.

   Important
   We urge you to restrict access to your `life-cycle-configuration` folder to Studio administrators and to your EMR Studio service role, and that you secure `configuration.json` against unauthorized access. For instructions, see Controlling access to a bucket with user policies or Security Best Practices for Amazon S3.

   For upload instructions, see Creating a folder and Uploading objects in the *Amazon Simple Storage Service User Guide*. To apply your configuration to an existing Workspace, close and restart the Workspace after you upload your configuration file to Amazon S3.

Optimize Spark jobs in EMR Studio

When running a Spark job using EMR Studio, there are a few steps you can take to help ensure that you're optimizing your Amazon EMR cluster resources.

Prolong your Livy session

If you use Apache Livy along with Spark on your Amazon EMR cluster, we recommend that you increase your Livy session timeout by doing one of the following:

- When you create an Amazon EMR cluster, set this configuration classification in the **Enter Configuration** field.

```
[
    {
        "Classification": "livy-conf",
        "Properties": {
          "livy.server.session.timeout": "8h"
        }
    }
]
```

- For an already-running EMR cluster, connect to your cluster using `ssh` and set the `livy-conf` configuration classification in `/etc/livy/conf/livy.conf`.

```
[
    {
        "Classification": "livy-conf",
        "Properties": {
          "livy.server.session.timeout": "8h"
        }
    }
]
```

  You may need to restart Livy after changing the configuration.
- If you don't want your Livy session to time out at all, set the property `livy.server.session.timeout-check` to `false` in `/etc/livy/conf/livy.conf`.

Run Spark in cluster mode

In cluster mode, the Spark driver runs on a core node instead of on the master node, improving resource utilization on the master node.

To run your Spark application in cluster mode instead of the default client mode, choose **Cluster** mode when you set **Deploy mode** while configuring your Spark step in your new Amazon EMR cluster. For more information, see Cluster mode overview in the Apache Spark documentation.

Increase Spark driver memory

To increase the Spark driver memory, configure your Spark session using the `%%configure` magic command in your EMR notebook, as in the following example.

```
%%configure -f
{"driverMemory": "6000M"}
```

Use an Amazon EMR Studio

In this section, you'll find topics that help users who are logged in to an Amazon EMR Studio configure and interact with the Studio.

The following video covers practical information such as how to create a new Workspace, and how to launch a new Amazon EMR cluster using a cluster template. The video also demonstrates running through a sample notebook.

This section includes the following topics to help you work in an EMR Studio:

- Learn Workspace basics (p. 82)
- Configure Workspace collaboration (p. 86)
- Run Workspace notebooks programmatically (p. 88)
- Browse data with SQL Explorer (p. 88)
- Attach a cluster to a Workspace (p. 89)
- Link Git-based repositories to an EMR Studio Workspace (p. 92)
- Debug applications and jobs with EMR Studio (p. 94)
- Install kernels and libraries in an EMR Studio Workspace (p. 97)
- Enhance kernels with magic commands (p. 97)
- Use multi-language notebooks with Spark kernels (p. 102)

Learn Workspace basics

When you use an EMR Studio, you can create and configure different *Workspaces* to organize and run notebooks. This section covers creating and working with Workspaces. For a conceptual overview, see Workspaces (p. 30) on the How Amazon EMR Studio works (p. 26) page.

This section covers the following topics to help you use EMR Studio Workspaces:

- Create an EMR Studio Workspace (p. 82)
- Launch a Workspace (p. 83)
- Understand the Workspace user interface (p. 83)
- Explore notebook examples (p. 84)
- Save Workspace content (p. 84)
- Delete a Workspace and notebook files (p. 84)
- Understand Workspace status (p. 85)
- Resolve Workspace connectivity issues (p. 86)

Create an EMR Studio Workspace

You can create EMR Studio Workspaces to run notebook code using the EMR Studio interface.

To create a Workspace in an EMR Studio

1. Log in to your EMR Studio.
2. Choose **Create Workspace**.
3. Enter a **Workspace name** and a **Description**. Naming a Workspace helps you identify it on the **Workspaces** page.
4. (Optional) Enable Workspace collaboration if you want to work with other Studio users in this Workspace in real time. You configurare collaborators after you launch the Workspace.

5. (Optional) To attach a cluster to a Workspace when you create the Workspace, expand the **Advanced configuration** section.

   Note
   Provisioning a new cluster requires access permissions from your administrator.

   Choose one of the cluster options for the Workspace and attach the cluster. For more information about provisioning a cluster when you create a Workspace, see Create a new EMR cluster (p. 90).
6. Choose **Create Workspace** in the lower right of the page.

After you create a Workspace, EMR Studio will open the **Workspaces** page. You will see a green success banner at the top of the page and can find the newly-created Workspace in the list.

By default, a Workspace is shared and can be seen by all Studio users. However, only one user can open and work in a Workspace at a time. To work simultaneously with other users, you can Configure Workspace collaboration (p. 86)

Launch a Workspace

To start working with notebook files, launch a Workspace to access the notebook editor. The **Workspaces** page in a Studio lists all of the Workspaces that you have access to with details including **Name**, **Status**, **Creation time**, and **Last modified**.

To launch a Workspace for editing and running notebooks

1. On the **Workspaces** page of your Studio, find the Workspace. You can filter the list by keyword or by column value.
2. Choose the Workspace name to launch the Workspace in a new browser tab. It may take a few minutes for the Workspace to open if it's **Idle**.

   Note
   Only one user can open and work in a Workspace at a time. If you select a Workspace that is already in use, EMR Studio displays a notification when you try to open it. The **User** column on the **Workspaces** page shows the user working in the Workspace.

Understand the Workspace user interface

The EMR Studio Workspace user interface is based on the JupyterLab interface with icon-denoted tabs on the left sidebar. When you pause over an icon, you can see a tooltip that shows the name of the tab. Choose tabs from the left sidebar to access the following panels.

- **File Browser** – Displays the files and directories in the Workspace, as well as the files and directories of linked Git repositories.
- **Running Kernels and Terminals** – Lists all of the kernels and terminals running in the Workspace. For more information, see Managing kernels and terminals in the official JupyterLab documentation.
- **Git** – Provides a graphical user interface for performing commands in the Git repositories attached to the Workspace. This panel is a JupyterLab extension called jupyterlab-git. For more information, see jupyterlab-git.
- **EMR Clusters** – Lets you attach a cluster to or detach a cluster from the Workspace to run notebook code. The EMR cluster configuration panel also provides advanced configuration options to help you create and attach a *new* cluster to the Workspace. For more information, see Create a new EMR cluster (p. 90).
- **EMR Git Repository** – Helps you link the Workspace with up to three Git repositories. For details and instructions, see Link Git-based repositories to an EMR Studio Workspace (p. 92).

- **Notebook Examples** – Provides a list of notebook examples that you can save to the Workspace. You can also access the examples by choosing **Notebook Examples** on the **Launcher** page of the Workspace.
- **Commands** – Offers a keyboard-driven way to search for and run JupyterLab commands. For more information, see the Command palette page in the JupyterLab documentation.
- **Notebook Tools** – Lets you select and set options such as cell slide type and metadata. The **Notebook Tools** option appears in the left sidebar after you open a notebook file.
- **Open Tabs** – Lists the open documents and activities in the main work area so that you can jump to an open tab. For more information, see the Tabs and single-document mode page in the JupyterLab documentation.
- **Collaboration** – Lets you enable or disable Workspace collaboration, and manage collaborators. To see the **Collaboration** panel, you must have the necessary permissions. For more information, see Set ownership for Workspace collaboration (p. 48).

Explore notebook examples

Every EMR Studio Workspace includes a set of notebook examples that you can use to explore EMR Studio features. To edit or run a notebook example, you can save it to the Workspace.

To save a notebook example to a Workspace

1. From the left sidebar, choose the **Notebook Examples** tab to open the **Notebook Examples** panel. You can also access the examples by choosing **Notebook Examples** on the **Launcher** page of the Workspace.
2. Choose a notebook example to preview it in the main work area. The example is read-only.
3. To save the notebook example to the Workspace, choose **Save to Workspace**. EMR Studio saves the example in your home directory. After you save a notebook example to the Workspace, you can rename, edit, and run it.

For more information about the notebook examples, see the EMR Studio Notebook examples GitHub repository.

Save Workspace content

When you work in the notebook editor of a Workspace, EMR Studio saves the content of notebook cells and output for you in the Amazon S3 location associated with the Studio. This backup process preserves work between sessions.

You can also save a notebook by pressing **CTRL+S** in the open notebook tab or by using one of the save options under **File**.

Another way to back up the notebook files in a Workspace is to associate the Workspace with a Git-based repository and sync your changes with the remote repository. Doing so also lets you save and share notebooks with team members who use a different Workspace or Studio. For instructions, see Link Git-based repositories to an EMR Studio Workspace (p. 92).

Delete a Workspace and notebook files

When you delete a notebook file from an EMR Studio Workspace, you delete the file from the **File browser**, and EMR Studio removes its backup copy in Amazon S3. You do not have to take any further steps to avoid storage charges when you delete a file from a Workspace.

When you delete *an entire Workspace*, EMR Studio does not remove any corresponding notebook files and folders in Amazon S3. These files remain in Amazon S3 and continue to accrue storage charges.

To avoid storage charges, you must remove all backed-up files and folders associated with a deleted Workspace from Amazon S3.

To delete a notebook file from an EMR Studio Workspace

1. Select the **File browser** panel from the left sidebar in the Workspace.
2. Select the file or folder you want to delete. Right-click your selection and choose **Delete**. The file disappears from the list and can no longer be opened. EMR Studio removes the file or folder from Amazon S3 for you.

To delete a Workspace and its associated backup files from EMR Studio

1. Log in to your EMR Studio with your Studio access URL and choose **Workspaces** from the left navigation.
2. Find your Workspace in the list, then select the check box next to its name. You can select multiple Workspaces to delete at the same time.
3. Choose **Delete** in the upper right of the **Workspaces** list and confirm that you want to delete the selected Workspaces. Choose **Delete** to confirm.
4. Follow the instructions for Deleting objects in the *Amazon Simple Storage Service Console User Guide* to remove the notebook files associated with the deleted Workspace from Amazon S3. If you did not create the Studio, consult your Studio administrator to determine the Amazon S3 backup location for the deleted Workspace.

Understand Workspace status

After you create an EMR Studio Workspace, it appears as a row in the **Workspaces** list in your Studio with its name, status, creation time, and last modified timestamp. The following table describes Workspace statuses.

Status	Description
Starting	The Workspace is being prepared, but is not yet ready to use. You can't open a Workspace when its status is Starting.
Ready	You can open the Workspace to use the notebook editor, but you must attach the Workspace to an EMR cluster before you can run notebook code.
Attaching	The Workspace is being attached to a cluster.
Attached	The Workspace is attached to an EMR cluster and ready for you to write and run notebook code. If a Workspace's status is not **Attached**, you must attach it to a cluster before you can run notebook code.
Idle	The Workspace is stopped and idle. To reactivate an idle Workspace, select it from the Workspaces list. The status changes from **Idle** to **Starting** to **Ready** when you select the Workspace.
Stopping	The Workspace is being stopped and will be set to **Idle**. EMR Studio stops notebooks that have been inactive for a long time.

Status	Description
Deleting	When you delete a Workspace, EMR Studio marks it for deletion and starts the deletion process. After the deletion process completes, the Workspace disappears from the list.

Resolve Workspace connectivity issues

To resolve Workspace connectivity issues, you can stop and restart a Workspace. When you restart a Workspace, EMR Studio launches the Workspace in a different Availability Zone or a different subnet that is associated with your Studio.

To stop and restart an EMR Studio Workspace

1. Close the Workspace in your browser.
2. Navigate to the **Workspace** list in the EMR Studio UI.
3. Select your Workspace from the list and choose **Actions**.
4. Choose **Stop** and wait for the Workspace status to change from **Stopping** to **Idle**.
5. Choose **Actions** again, and then choose **Start** to restart the Workspace.
6. Wait for the Workspace status to change from **Starting** to **Ready**, then choose the Workspace name to reopen it in a new browser tab.

Configure Workspace collaboration

Workspace collaboration lets you write and run notebook code simultaneously with other members of your team. When you work in the same notebook file, you'll see changes as your collaborators make them. You can enable collaboration when you create a Workspace, or switch collaboration on and off in an existing Workspace.

Prerequisites

Before you configure collaboration for a Workspace, make sure you complete the following tasks:

- Ensure that your EMR Studio admin has given you the necessary permissions. For example, the following statement allows a user to configure collaboration for any Workspace with the tag key `creatorUserId` whose value matches the user's ID (indicated by the policy variable `aws:userId`).

```
{
    "Sid": "UserRolePermissionsForCollaboration",
    "Action": [
        "elasticmapreduce:UpdateEditor",
        "elasticmapreduce:PutWorkspaceAccess",
        "elasticmapreduce:DeleteWorkspaceAccess",
        "elasticmapreduce:ListWorkspaceAccessIdentities"
    ],
    "Resource": "*",
    "Effect": "Allow",
    "Condition": {
        "StringEquals": {
            "elasticmapreduce:ResourceTag/creatorUserId": "${aws:userid}"
        }
    }
}
```

- Ensure that the service role associated with your EMR Studio has the permissions required to enable and configure Workspace collaboration, as in the following example statement.

```
{
    "Sid": "AllowWorkspaceCollaboration",
    "Effect": "Allow",
    "Action": [
        "iam:GetUser",
        "iam:GetRole",
        "iam:ListUsers",
        "iam:ListRoles",
        "sso:GetManagedApplicationInstance",
        "sso-directory:SearchUsers"
    ],
    "Resource": "*"
}
```

For more information, see Create an EMR Studio service role (p. 42).

To enable Workspace collaboration and add collaborators

1. In your Workspace, choose the **Collaboration** icon from the Launcher screen or the bottom of the left panel.

   Note
   You won't see the **Collaboration** panel unless your Studio administator has given you permission to configure collaboration for the Workspace. For more information, see Set ownership for Workspace collaboration (p. 48).

2. Make sure the **Allow Workspace collaboration** toggle is in the on position. When you enable collaboration, only you and the collaborators that you add can see the Workspace in the list on the Studio **Workspaces** page.
3. Enter a **Collaborator name**. Your Workspace can have a maximum of five collaborators including yourself. A collaborator can be any user with access to your EMR Studio. If you don't enter a collaborator, the Workspace is a private Workspace that is only accessible to you.

   The following table specifies the applicable collaborator values to enter based on the identity type of the owner.

   Note
   An owner can only invite collaborators with the same identity type. For example, an IAM user can only add other IAM users, and an IAM Identity Center user can only add other IAM Identity Center users.

Authentication mode	Value to enter for Collaborator name
IAM authentication	An IAM username. This is the name that a user sees when logged in to the AWS Management Console.
IAM federation	The name of an IAM role and an optional session name. To add all of the federated users who assume the same IAM role, specify the name of an IAM role for federation.

Authentication mode	Value to enter for Collaborator name
	To add a single user as a collaborator, specify a role and session name. For example, `MyRoleName:MySessionName`.
SSO	An IAM Identity Center user name like `user@example.com`.

4. Choose **Add**. The collaborator can now see the Workspace on their EMR Studio **Workspaces** page, and launch the Workspace to use it in real time with you.

   Note
   If you disable Workspace collaboration, the Workspace returns to its shared state and can be seen by all Studio users. In the shared state, only one Studio user can open and work in the Workspace at a time.

Run Workspace notebooks programmatically

You can run your Amazon EMR Studio Workspace notebooks programmatically with a script or on the AWS CLI. To learn how to run your notebook programatically, see Sample commands to execute EMR Notebooks programmatically (p. 111).

Browse data with SQL Explorer

This topic provides information to help you get started with Amazon EMR Studio's SQL Explorer. SQL Explorer is a single-page tool in your Workspace that helps you understand the data sources in your EMR cluster's data catalog. You can use SQL Explorer to browse your data, run SQL queries to retrieve data, and download query results.

SQL Explorer supports Presto. Before you use SQL Explorer, make sure you have a cluster that uses Amazon EMR version 5.34.0 or later or version 6.4.0 or later with Presto installed. The Amazon EMR Studio SQL Explorer doesn't support Presto clusters that you've configured with in-transit encryption. This is because Presto runs in TLS mode on these clusters.

Browse your cluster's data catalog

SQL Explorer provides a catalog browser interface that you can use to explore and understand how your data is organized. For example, you can use the data catalog browser to verify table and column names before you write a SQL query.

To browse your data catalog

1. Open SQL Explorer in your Workspace.
2. Make sure your Workspace is attached to an EMR cluster running on EC2 that uses Amazon EMR version 6.4.0 or later with Presto installed. You can choose an existing cluster, or create a new one. For more information, see Attach a cluster to a Workspace (p. 89).
3. Select a **Database** from the dropdown list to browse.
4. Expand a table in your database to see the table's column names. You can also enter a keyword in the search bar to filter table results.

Run a SQL query to retrieve data

To retrieve data with a SQL query and download the results

1. Open SQL Explorer in your Workspace.
2. Make sure your Workspace is attached to an EMR cluster running on EC2 with Presto and Spark installed. You can choose an existing cluster, or create a new one. For more information, see Attach a cluster to a Workspace (p. 89).
3. Select **Open editor** to open a new editor tab in your Workspace.
4. Compose your SQL query in the editor tab.
5. Choose **Run**.
6. View your query results under **Result preview**. SQL Explorer displays the first 100 results by default. You can choose a different number of results to display (up to 1000) using the **Preview first 100 query results** drowdown.
7. Choose **Download results** to download your results in CSV format. You can download up to 1000 rows of results.

Attach a cluster to a Workspace

Amazon EMR Studio runs notebook commands using a kernel on an EMR cluster. Before you can select a kernel, you should attach the Workspace to a cluster that uses Amazon EC2 instances, or to an EMR on EKS cluster. EMR Studio lets you attach Workspaces to new or existing clusters, and gives you the flexibility to change clusters without closing the Workspace.

This section covers the following topics to help you work with and provision clusters for EMR Studio:

- Attach a running cluster (p. 89)
- Use an Amazon EMR on EKS cluster (p. 90)
- Create a new EMR cluster (p. 90)
- Detach a cluster (p. 92)

Attach a running cluster

You can attach an existing EMR cluster running on Amazon EC2 to a Workspace when you create the Workspace, or you can choose a cluster from the Workspace user interface. If you want to create and attach a *new* cluster, see Create a new EMR cluster (p. 90).

Create a Workspace dialog box

To attach a running cluster when you create a Workspace

1. In the **Create a Workspace** dialog box, make sure you've already selected a subnet for the new Workspace. Expand the **Advanced configuration** section.
2. Choose **Attach Workspace to an EMR cluster**.
3. In the **EMR cluster** dropdown list, select an existing EMR cluster to attach it to the Workspace.

After you attach a cluster, you can finish the Workspace creation process. When you open the new Workspace for the first time and choose the **EMR Clusters** panel, you should see that your selected cluster is attached.

Workspace UI

To attach a running cluster from the Workspace user interface

1. In the Workspace that you want to attach to a cluster, choose the **EMR Clusters** icon from the left sidebar to open the **Cluster** panel.
2. Under **Cluster type**, expand the dropdown and select **EMR Cluster on EC2**.
3. Choose a cluster from the dropdown list. You might need to detach an existing cluster first to enable the cluster selection dropdown list.
4. Choose **Attach**. When the cluster is attached, you should see a success message appear.

Use an Amazon EMR on EKS cluster

In addition to using Amazon EMR clusters running on Amazon EC2, you can attach a Workspace to an Amazon EMR on EKS cluster to run notebook code. For more information about EMR on EKS, see What is Amazon EMR on EKS.

Before you can connect a Workspace to an EMR on EKS cluster, your Studio administrator must grant you access permissions.

Create a Workspace dialog box

To attach an EMR on EKS cluster when you create a Workspace

1. In the **Create a Workspace** dialog box, expand the **Advanced configuration** section.
2. Choose **Attach Workspace to an EMR on EKS cluster**.
3. Under **EMR on EKS cluster**, choose a cluster from the dropdown list.
4. Under **Select an endpoint**, choose a managed endpoint to attach to the Workspace. A managed endpoint is a gateway that lets EMR Studio communicate with your chosen cluster.
5. Choose **Create Workspace** to finish the Workspace creation process and attach the selected cluster.

After you attach a cluster, you can finish the Workspace creation process. When you open the new Workspace for the first time and choose the **EMR Clusters** panel, you should see that your selected cluster is attached.

Workspace UI

To attach an EMR on EKS cluster from the Workspace user interface

1. In the Workspace that you want to attach to a cluster, choose the **EMR Clusters** icon from the left sidebar to open the **Cluster** panel.
2. Expand the **Cluster type** dropdown and choose **EMR clusters on EKS**.
3. Under **EMR cluster on EKS**, choose a cluster from the dropdown list.
4. Under **Endpoint**, choose a managed endpoint to attach to the Workspace. A managed endpoint is a gateway that lets EMR Studio communicate with your chosen cluster.
5. Choose **Attach**. When the cluster is attached, you should see a success message appear.

Create a new EMR cluster

Advanced EMR Studio users can provision new EMR clusters running on Amazon EC2 to use with a Workspace. The new cluster has all of the big data applications that are required for EMR Studio installed by default.

To create clusters, your Studio administrator must first give you permission using a session policy. For more information, see Create permissions policies for EMR Studio users (p. 48).

You can create a new cluster in the **Create a Workspace** dialog box or from the **Cluster** panel in the Workspace UI. Either way, you have two cluster creation options:

1. **Create an EMR cluster** – Create an EMR cluster by choosing the Amazon EC2 instance type and count.
2. **Use a cluster template** – Provision a cluster by selecting a predefined cluster template. This option appears if you have permission to use cluster templates.

To create an EMR cluster by providing a cluster configuration

1. Choose a starting point.

To...	Do this...
Create the cluster when you create a Workspace with the **Create a Workspace** dialog box.	Expand the **Advanced configuration** section in the **Create a Workspace** dialog box, and select **Create an EMR cluster**.
Create the cluster from the EMR Cluster panel in the Workspace UI after you have created a Workspace.	Choose the **EMR Clusters** tab in the left sidebar of an open Workspace, expand the **Advanced configuration** section, and choose **Create cluster**.

2. Enter a **Cluster name**. Naming the cluster helps you find it later in the EMR Studio Clusters list.
3. For **EMR release**, Choose an EMR release version for the cluster.
4. For **Instance**, select the type and number of Amazon EC2 instances for the cluster. For more information about selecting instance types, see Configure Amazon EC2 instances (p. 197). One instance will be used as the master node.
5. Select a **Subnet** where EMR Studio can launch the new cluster. Each subnet option is preapproved by your Studio administrator, and your Workspace should be able to connect to a cluster in any listed subnet.
6. Choose an **S3 URI for log storage**.
7. Choose **Create EMR cluster** to provision the cluster. If you use the **Create a Workspace** dialog box, choose **Create Workspace** to create the Workspace and provision the cluster. After EMR Studio provisions the new cluster, it attaches the cluster to the Workspace.

To create a cluster using a cluster template

1. Choose a starting point.

To...	Do this...
Create the cluster when you create a Workspace with the **Create a Workspace** dialog box.	Expand the **Advanced configuration** section in the **Create a Workspace** dialog box, and select **Use a cluster template**.
Create the cluster from the EMR Cluster panel in the Workspace UI.	Choose the **EMR Clusters** tab in the left sidebar of an open Workspace, expand the **Advanced configuration** section, then choose **Cluster template**.

2. Select a cluster template from the dropdown list. Each available cluster template includes a brief description to help you make a selection.

3. The cluster template you choose may have additional parameters such as Amazon EMR release version or cluster name. You can choose or insert values, or use the default values that your administrator selected.
4. Select a **Subnet** where EMR Studio can launch the new cluster. Each subnet option is preapproved by your Studio administrator, and your Workspace should be able to connect to a cluster in any subnet.
5. Choose **Use cluster template** to provision the cluster and attach it to the Workspace. It will take a few minutes for EMR Studio to create the cluster. If you use the **Create a Workspace** dialog box, choose **Create Workspace** to create the Workspace and provision the cluster. After EMR Studio provisions the new cluster, it attaches the cluster to your Workspace.

Detach a cluster

To exchange the cluster attached to a Workspace, you can detach a cluster from the Workspace UI.

To detach a cluster from a Workspace

1. In the Workspace that you want to detach from a cluster, choose the **EMR Clusters** icon from the left sidebar to open the **Cluster** panel.
2. Under **Select cluster**, choose **Detach** and wait for EMR Studio to detach the cluster. When the cluster is detached, you will see a success message.

Link Git-based repositories to an EMR Studio Workspace

About Git repositories for EMR Studio

You can associate a maximum of three Git repositories with an EMR Studio Workspace. By default, each Workspace lets you choose from a list of Git repositories that are associated with the same AWS account as the Studio. You can also create a new Git repository as a resource for a Workspace.

You can run Git commands like the following using a terminal command while connected to the master node of a cluster.

```
!git pull origin <branch-name>
```

Alternatively, you can use he jupyterlab-git extension. Open it from the left sidebar by choosing the **Git** icon. For information about the jupyterlab-git extension for JupyterLab, see jupyterlab-git.

Prerequisites

- To associate a Git repository with a Workspace, the Studio must be configured to allow Git repository linking. Your Studio administrator should take steps to Establish access and permissions for Git-based repositories (p. 79).
- If you use a CodeCommit repository, you must use Git credentials and HTTPS. SSH keys and HTTPS with the AWS Command Line Interface credential helper are not supported. CodeCommit also does not support personal access tokens (PATs). For more information, see Using IAM with CodeCommit in the *IAM User Guide* and Setup for HTTPS users using Git credentials in the *AWS CodeCommit User Guide*.

Instructions

To link an associated Git repository to a Workspace

1. Open the Workspace that you want to link to a repository from the **Workspaces** list in the Studio.
2. In the left sidebar, choose the **EMR Git Repository** icon to open the **Git repository** tool panel.
3. Under **Git repositories**, expand the dropdown list and select a maximum of three repositories to link to the Workspace. EMR Studio registers your selection and begins linking each repository.

It might take some time for the linking process to complete. You can see the status for each repository that you selected in the **Git repository** tool panel. After EMR Studio links a repository to a Workspace, you should see the files that belong to that repository appear in the **File browser** panel.

To add a new Git repository to a Workspace as a resource

1. Open the Workspace that you want to link to a repository from the Workspaces list in your Studio.
2. In the left sidebar, choose the **EMR Git Repository** icon to open the **Git repository** tool panel.
3. Choose **Add new Git repository**.
4. For **Repository name**, enter a descriptive name for the repository in EMR Studio. Names may only contain alphanumeric characters, hyphens, and underscores.
5. For **Git repository URL**, enter the URL for the repository. When you use a CodeCommit repository, this is the URL that is copied when you choose **Clone URL** and then **Clone HTTPS**. For example, `https://git-codecommit.us-west-2.amazonaws.com/v1/repos/[MyCodeCommitRepoName]`.
6. For **Branch**, enter the name of an existing branch that you want to check out.
7. For Git credentials, choose an option according to the following guidelines. EMR Studio accesses your Git credentials using secrets stored in Secrets Manager.

   Note
   If you use a GitHub repository, we recommend that you use a personal access token (PAT) to authenticate. Beginning August 13, 2021, GitHub will require token-based authentication and will no longer accept passwords when authenticating Git operations. For more information, see the Token authentication requirements for Git operations post in *The GitHub Blog*.

Option	Description
Create a new secret	Choose this option to associate existing Git credentials with a new secret that will be created in AWS Secrets Manager for you. Do one of the following based on the Git credentials that you use for the repository. If you use a Git user name and password to access the repository, select **Username and password**, enter the **Secret name** to use in Secrets Manager, and then enter the **Username** and **Password** to associate with the secret. –OR– If you use a personal access token to access the repository, select **Personal access token (PAT)**, enter the **Secret name** to use in Secrets Manager, and then enter your **personal access**

Option	Description
	token. For more information, see Creating a personal access token for the command line for GitHub and Personal access tokens for Bitbucket. CodeCommit repositories do not support this option.
Use a public repository without credentials	Choose this option to access a public repository.
Use an existing AWS secret	Choose this option if you already saved your credentials as a secret in Secrets Manager, and then select the secret name from the list. If you select a secret associated with a Git user name and password, the secret must be in the format `{"gitUsername": "MyUserName", "gitPassword": "MyPassword"}`.

8. Choose **Add repository** to create the new repository. After EMR Studio creates the new repository, you will see a success message. The new repository appears in the dropdown list under **Git repositories**.
9. To link the new repository to your Workspace, choose it from the dropdown list under **Git repositories**.

It might take some time for the linking process to complete. After EMR Studio links the new repository to the Workspace, you should see a new folder with the same name as your repository appear in the **File Browser** panel.

To open a different linked repository, navigate to its folder in the **File browser**.

Debug applications and jobs with EMR Studio

With Amazon EMR Studio, you can launch data application interfaces to analyze applications and job runs in the browser.

You can also launch the persistent, off-cluster user interfaces for Amazon EMR running on EC2 clusters from the Amazon EMR console. For more information, see View persistent application user interfaces (p. 558).

> **Note**
> Depending on your browser settings, you might need to enable pop-ups for an application UI to open.

For information about configuring and using the application interfaces, see The YARN Timeline Server, Monitoring and instrumentation, or Tez UI overview.

Debug Amazon EMR running on Amazon EC2 jobs

Workspace UI

Launch an on-cluster UI from a notebook file

When you use Amazon EMR release versions 5.33.0 and later, you can launch the Spark web user interface (the Spark UI or Spark History Server) from a notebook in your Workspace.

On-cluster UIs work with the PySpark, Spark, or SparkR kernels. The maximum viewable file size for Spark event logs or container logs is 10 MB. If your log files exceed 10 MB, we recommend that you use the persistent Spark History Server instead of the on-cluster Spark UI to debug jobs.

Important

In order for EMR Studio to launch on-cluster application user interfaces from a Workspace, a cluster must be able to communicate with the Amazon API Gateway. You must configure the EMR cluster to allow outgoing network traffic to Amazon API Gateway, and make sure that Amazon API Gateway is reachable from the cluster.

The Spark UI accesses container logs by resolving hostnames. If you use a custom domain name, you must make sure that the hostnames of your cluster nodes can be resolved by Amazon DNS or by the DNS server you specify. To do so, set the Dynamic Host Configuration Protocol (DHCP) options for the Amazon Virtual Private Cloud (VPC) that is associated with your cluster. For more information about DHCP options, see DHCP option sets in the *Amazon Virtual Private Cloud User Guide.*

1. In your EMR Studio, open the Workspace that you want to use and make sure that it is attached to an Amazon EMR cluster running on EC2. For instructions, see Attach a cluster to a Workspace (p. 89).
2. Open a notebook file and use the PySpark, Spark, or SparkR kernel. To select a kernel, choose the kernel name from the upper right of the notebook toolbar to open the **Select Kernel** dialog box. The name appears as **No Kernel!** if no kernel has been selected.
3. Run your notebook code. The following appears as output in the notebook when you start the Spark context. It might take a few seconds to appear. If you have started the Spark context, you can run the `%%info` command to access a link to the Spark UI at any time.

   Note

   If the Spark UI links do not work or do not appear after a few seconds, create a new notebook cell and run the `%%info` command to regenerate the links.

4. To launch the Spark UI, choose **Link** under **Spark UI**. If your Spark application is running, the Spark UI opens in a new tab. If the application has completed, the Spark History Server opens instead.

   After you launch the Spark UI, you can modify the URL in the browser to open the YARN ResourceManager or the Yarn Timeline Server. Add one of the following paths after `amazonaws.com`.

Web UI	Path	Example modified URL
YARN ResourceManager	/rm	https://*j-examplebby5ij*.emrappui-prod.*eu-west-1*.amazonaws.com/*rm*
Yarn Timeline Server	/yts	https://*j-examplebby5ij*.emrappui-prod.*eu-west-1*.amazonaws.com/*yts*
Spark History Server	/shs	https://*j-examplebby5ij*.emrappui-prod.*eu-west-1*.amazonaws.com/*shs*

Studio UI

Launch the persistent YARN Timeline Server, Spark History Server, or Tez UI from the EMR Studio UI

1. In your EMR Studio, select **EMR on EC2** on the left side of the page to open the **EMR on EC2** clusters list.
2. Filter the list of clusters by **name**, **state**, or **ID** by entering values in the search box. You can also search by creation **time range**.
3. Select a cluster and then choose **Launch application UIs** to select an application user interface. The Application UI opens in a new browser tab and might take some time to load.

Debug Amazon EMR on EKS job runs with the Spark History Server

When you submit a job run to an Amazon EMR on EKS cluster, you can access logs for that job run using the Spark History Server. The Spark History Server provides tools for monitoring Spark applications, such as a list of scheduler stages and tasks, a summary of RDD sizes and memory usage, and environmental information. You can launch the Spark History Server for Amazon EMR on EKS job runs in the following ways:

- When you submit a job run using EMR Studio with an Amazon EMR on EKS managed endpoint, you can launch the Spark History Server from a notebook file in your Workspace.
- When you submit a job run using the AWS CLI or AWS SDK for Amazon EMR on EKS, you can launch the Spark History Server from the EMR Studio UI.

For information about how to use the Spark History Server, see Monitoring and Instrumentation in the Apache Spark documentation. For more information about job runs, see Concepts and components in the *Amazon EMR on EKS Development Guide*.

To launch the Spark History Server from a notebook file in your EMR Studio Workspace

1. Open a Workspace that is connected to an Amazon EMR on EKS cluster.
2. Select and open your notebook file in the Workspace.
3. Choose **Spark UI** at the top of the notebook file to open the persistent Spark History Server in a new tab.

To launch the Spark History Server from the EMR Studio UI

Note
The **Jobs** list in the EMR Studio UI displays only job runs that you submit using the AWS CLI or AWS SDK for Amazon EMR on EKS.

1. In your EMR Studio, select **EMR on EKS** on the left side of the page.
2. Search for the Amazon EMR on EKS virtual cluster that you used to submit your job run. You can filter the list of clusters by **status** or **ID** by entering values in the search box.
3. Select the cluster to open its detail page. The detail page displays information about the cluster, such as ID, namespace, and status. The page also shows a list of all the job runs submitted to that cluster.
4. From the cluster detail page, select a job run to debug.
5. In the upper right of the **Jobs** list, choose **Launch Spark History Server** to open the application interface in a new browser tab.

Install kernels and libraries in an EMR Studio Workspace

Each Amazon EMR Studio Workspace comes with a set of pre-installed libraries and kernels. You can also customize the environment for EMR Studio in the following ways when you use EMR clusters running on Amazon EC2:

- **Install Jupyter Notebook kernels and Python libraries on a cluster master node** – When you install libraries using this option, all Workspaces attached to the same cluster share those libraries. You can install kernels or libraries from within a notebook cell or while connected using SSH to the master node of a cluster.
- **Use notebook-scoped libraries** – When Workspace users install and use libraries from within a notebook cell, those libraries only available to that notebook alone. This option lets different notebooks using the same cluster work without worrying about conflicting library versions.

EMR Studio Workspaces have the same underlying architecture as EMR notebooks. You can install and use Jupyter Notebook kernels and Python libraries with EMR Studio in the same way you would with EMR notebooks. For instructions, see Installing and using kernels and libraries (p. 121).

Kernels and libraries on Amazon EMR on EKS clusters

Amazon EMR on EKS clusters include the PySpark and Python 3.7 kernels with a set of pre-installed libraries. Amazon EMR on EKS does not support installing additional libraries or clusters.

Each Amazon EMR on EKS cluster comes with the following Python and PySpark libraries installed:

- **Python**– boto3, cffi, future, ggplot, jupyter, kubernetes, matplotlib, numpy, pandas, plotly, pycryptodomex, py4j, requests, scikit-learn, scipy, seaborn
- **PySpark** – ggplot, jupyter, matplotlib, numpy, pandas, plotly, pycryptodomex, py4j, requests, scikit-learn, scipy, seaborn

Enhance kernels with magic commands

EMR Studio and EMR Notebooks support magic commands, which are enhancements provided by the IPython kernel to help run and analyze data. IPython is an interactive shell environment built with Python.

Amazon EMR also supports Sparkmagic, a package that provides Spark-related kernels (PySpark, SparkR, and Scala kernels) with specific magic commands and that uses Livy on the cluster to submit Spark jobs.

You can use magic commands as long as you have a Python kernel in your EMR notebook. Similarly, any Spark-related kernel supports Sparkmagic commands.

Magic commands, also called *magics*, come in two varieties:

- **Line magics** – These magic commands are denoted by a single % prefix and operate on a single line of code
- **Cell magics** – These magic commands are denoted by a double %% prefix and operate on multiple lines of code

A full list of magic commands can be found on the IPython website. To learn more about Sparkmagic commands, see Sparkmagic on the GitHub website.

Considerations and limitations

Amazon EMR on EKS clusters do not support Sparkmagic commands for EMR Studio. This is because Spark kernels used with managed endpoints are built into Kubernetes and are not supported by Sparkmagic and Livy. You can set the Spark configuration directly into the SparkContext object as a workaround, as the following example demonstrates.

```
spark.conf.set("spark.driver.maxResultSize", '6g')
```

AWS doesn't allow the following magic commands because of security concerns:

- `%alias`
- `%alias_magic`
- `%automagic`
- `%macro`
- Modifying `proxy_user` using `%configure`
- Modifying `KERNEL_USERNAME` using `%env` or `%set_env`

List magic and Sparkmagic commands

Use the following commands to list the available magic commands:

- `%lsmagic` lists all currently-available magic functions
- `%%help` lists currently-available Spark-related magic functions provided by the Sparkmagic package

Configure Spark with `%%configure`

One of the most useful Sparkmagic commands is the `%%configure` command, which configures the session creation parameters. Using `conf` settings, you can configure any Spark configuration mentioned in Spark's configuration documentation.

Example Add external JAR file to EMR Notebooks from Maven repository or Amazon S3

You can use the following approach to add an external JAR file dependency to any Spark-related kernel that's supported by Sparkmagic.

```
%%configure -f
{"conf": {
    "spark.jars.packages": "com.jsuereth:scala-arm_2.11:2.0,ml.combust.bundle:bundle-
ml_2.11:0.13.0,com.databricks:dbutils-api_2.11:0.0.3",
    "spark.jars": "s3://DOC-EXAMPLE-BUCKET/my-jar.jar"
    }
}
```

Example : Configure Hudi

You can use the notebook editor to configure your EMR notebook to use Hudi.

```
%%configure
{ "conf": {
     "spark.jars": "hdfs://apps/hudi/lib/hudi-spark-bundle.jar,hdfs:///apps/hudi/lib/spark-
spark-avro.jar",
```

```
     "spark.serializer": "org.apache.spark.serializer.KryoSerializer",
     "spark.sql.hive.convertMetastoreParquet":"false"
     }
}
```

Run Spark submit using %%sh

The %%sh magic runs shell commands in a subprocess on an instance of your attached cluster. Typically, you'd use one of the Spark-related kernels to run Spark applications on your attached cluster. However, if you want to use a Python kernel to submit a Spark application, you can use the following magic, replacing the bucket name with your bucket name in lowercase.

```
%%sh
spark-submit --master yarn --deploy-mode cluster s3://DOC-EXAMPLE-BUCKET/test.py
```

In this example, the cluster needs access to the location of s3://*DOC-EXAMPLE-BUCKET*/test.py, or the command will fail.

You can use any Linux command with the %%sh magic. If you want to run any Spark or YARN commands, use one of the following options to create an emr-notebook Hadoop user and grant the user permissions to run the commands:

- You can explicitly create a new user by running the following commands.

  ```
  hadoop fs -mkdir /user/emr-notebook
  hadoop fs -chown emr-notebook /user/emr-notebook
  ```

- You can turn on user impersonation in Livy, which automatically creates the user. See Enabling user impersonation to monitor Spark user and job activity (p. 118) for more information.

Visualize Spark dataframes using %%display

The %%display magic allows you to quickly visualize a dataframe. To use this magic, run the following command.

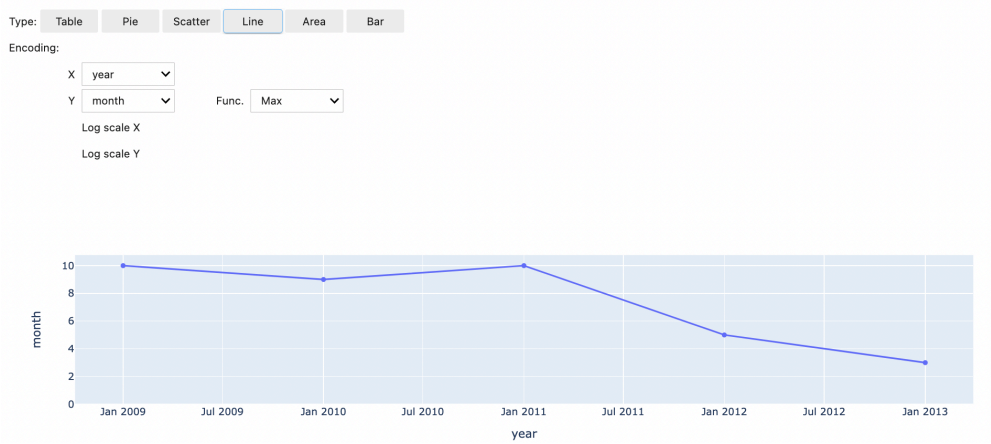
```
%%display df
```

You can choose to view the results in a table format, as the following image shows.

Type: Table Pie Scatter Line Area Bar

year	month	total_passengers	total_trips
2012-01-01	3	26866837	16146923
2011-01-01	3	26091246	16066350
2013-01-01	3	26965079	15749228
2011-01-01	10	26287953	15707756
2009-01-01	10	26202049	15604551
2012-01-01	5	26278817	15567525
2011-01-01	5	25508952	15554868
2010-01-01	9	25533166	15540209
2010-01-01	5	26002858	15481351
2012-01-01	4	25900645	15477914

You can also choose to visualize your data with five types of charts. Your options include pie, scatter, line, area, and bar charts.



Use EMR Notebooks magics

Amazon EMR provides the following EMR Notebooks magics that you can use with Python3 and Spark-based kernels:

- `%mount_workspace_dir` - Mounts your Workspace directory to your cluster so that you can import and run code from other files in your Workspace

  Note
  With `%mount_workspace_dir`, only the Python 3 kernel can access your local file systems. Spark executors will not have access to the mounted directory with this kernel.
- `%umount_workspace_dir` - Unmounts your Workspace directory from your cluster
- `%generate_s3_download_url` - Generates a temporary download link in your notebook output for an Amazon S3 object

Prerequisites

Before you install EMR Notebooks magics, complete the following tasks:

- Make sure that your Service role for cluster EC2 instances (EC2 instance profile) (p. 403) has read access for Amazon S3. The `EMR_EC2_DefaultRole` with the `AmazonElasticMapReduceforEC2Role` managed policy fulfills this requirement. If you use a custom role or policy, make sure that it has the necessary S3 permissions.

  Note
  EMR Notebooks magics run on a cluster as the notebook user and use the EC2 instance profile to interact with Amazon S3. When you mount a Workspace directory on an EMR cluster, all Workspaces and EMR notebooks with permission to attach to that cluster can access the mounted directory.
  Directories are mounted as read-only by default. While `s3fs-fuse` and `goofys` allow read-write mounts, we strongly recommend that you do not modify mount parameters to mount directories in read-write mode. If you allow write access, any changes made to the directory are written to the S3 bucket. To avoid accidental deletion or overwriting, you can enable versioning for your S3 bucket. To learn more, see Using versioning in S3 buckets.

- Run one of the following scripts on your cluster to install the dependencies for EMR Notebooks magics. To run a script, you can either Use custom bootstrap actions (p. 192) or follow the instructions in Run commands and scripts on an Amazon EMR cluster when you already have a running cluster.

  You can choose which dependency to install. Both s3fs-fuse and goofys are FUSE (Filesystem in Userspace) tools that let you mount an Amazon S3 bucket as a local file system on a cluster. The `s3fs` tool provides an experience similar to POSIX. The `goofys` tool is a good choice when you prefer performance over a POSIX-compliant file system.

```
#!/bin/sh

# Install the s3fs dependency for EMR Notebooks magics
sudo amazon-linux-extras install epel -y
sudo yum install s3fs-fuse -y
```

  OR

```
#!/bin/sh

# Install the goofys dependency for EMR Notebooks magics
sudo wget https://github.com/kahing/goofys/releases/latest/download/goofys -P /usr/bin/
sudo chmod ugo+x /usr/bin/goofys
```

Install EMR Notebooks magics

Complete the following steps to install EMR Notebooks magics.

1. In your notebook, run the following command to install the `emr-notebooks-magics` package.

```
%pip install emr-notebooks-magics
```

2. Restart your kernel to load the EMR Notebooks magics.
3. Verify your installation with the following command, which should display output help text for `%mount_workspace_dir`.

```
%mount_workspace_dir?
```

Mount a Workspace directory with `%mount_workspace_dir`

The `%mount_workspace_dir` magic lets you mount your Workspace directory onto your EMR cluster so that you can import and run other files, modules, or packages stored in your directory.

The following example mounts the entire Workspace directory onto a cluster, and specifies the optional *<--fuse-type>* argument to use goofys for mounting the directory.

```
%mount_workspace_dir . <--fuse-type goofys>
```

To verify that your Workspace directory is mounted, use the following example to display the current working directory with the `ls` command. The output should display all of the files in your Workspace.

```
%%sh
ls
```

When you're done making changes in your Workspace, you can unmount the Workspace directory with the following command.

> **Note**
> Your Workspace directory stays mounted to your cluster even when the Workspace is stopped or detached. You must explicitly unmount your Workspace directory.

```
%umount_workspace_dir
```

Download an Amazon S3 object with `%generate_s3_download_url`

The `generate_s3_download_url` command creates a presigned URL for an object stored in Amazon S3. You can use the presigned URL to download the object to your local machine. For example, you might run `generate_s3_download_url` to download the result of a SQL query that your code writes to Amazon S3.

The presigned URL is valid for 60 minutes by default. You can change the expiration time by specifying a number of seconds for the `--expires-in` flag. For example, `--expires-in 1800` creates a URL that is valid for 30 minutes.

The following example generates a download link for an object by specifying the full Amazon S3 path: *s3://EXAMPLE-DOC-BUCKET/path/to/my/object*.

```
%generate_s3_download_url s3://EXAMPLE-DOC-BUCKET/path/to/my/object
```

To learn more about using `generate_s3_download_url`, run the following command to display help text.

```
%generate_s3_download_url?
```

Use multi-language notebooks with Spark kernels

Each Jupyter notebook kernel has a default language. For example, the Spark kernel's default language is Scala, and the PySpark kernels's default language is Python. With Amazon EMR 6.4.0 and later, EMR Studio supports multi-language notebooks. This means that each kernel in EMR Studio can support the following languages in addition to the default language: Python, Spark, R, and Spark SQL.

To activate this feature, specify one of the following magic commands at the beginning of any cell.

Language	Command
Python	`%%pyspark`
Scala	`%%scalaspark`
R	`%%rspark`
Spark SQL	`%%sql`

When invoked, these commands execute the entire cell within the same Spark session using the interpreter of the corresponding language.

The `%%pyspark` cell magic allows users to write PySpark code in all Spark kernels.

```
%%pyspark
a = 1
```

The `%%sql` cell magic allows users to execute Spark-SQL code in all Spark kernels.

```
%%sql
SHOW TABLES
```

The `%%rspark` cell magic allows users to execute SparkR code in all Spark kernels.

```
%%rspark
a <- 1
```

The `%%scalaspark` cell magic allows users to execute Spark Scala code in all Spark kernels.

```
%%scalaspark
val a = 1
```

Share data across language interpreters with temporary tables

You can also share data between language interpreters using temporary tables. The following example uses `%%pyspark` in one cell to create a temporary table in Python and uses `%%scalaspark` in the following cell to read data from that table in Scala.

```
%%pyspark
df=spark.sql("SELECT * from nyc_top_trips_report LIMIT 20")
# create a temporary table called nyc_top_trips_report_view in python
df.createOrReplaceTempView("nyc_top_trips_report_view")
```

```
%%scalaspark
// read the temp table in scala
val df=spark.sql("SELECT * from nyc_top_trips_report_view")
df.show(5)
```

EMR Notebooks

You can use Amazon EMR Notebooks along with Amazon EMR clusters running Apache Spark to create and open Jupyter Notebook and JupyterLab interfaces within the Amazon EMR console. An EMR notebook is a "serverless" notebook that you can use to run queries and code. Unlike a traditional notebook, the contents of an EMR notebook itself—the equations, queries, models, code, and narrative text within notebook cells—run in a client. The commands are executed using a kernel on the EMR cluster. Notebook contents are also saved to Amazon S3 separately from cluster data for durability and flexible re-use.

You can start a cluster, attach an EMR notebook for analysis, and then terminate the cluster. You can also close a notebook attached to one running cluster and switch to another. Multiple users can attach notebooks to the same cluster simultaneously and share notebook files in Amazon S3 with each other. These features let you run clusters on-demand to save cost, and reduce the time spent re-configuring notebooks for different clusters and datasets.

You can also execute an EMR notebook programmatically using the EMR API, without the need to interact with EMR console ("headless execution"). You need to include a cell in the EMR notebook that has a parameters tag. That cell allows a script to pass new input values to the notebook. Parameterized notebooks can be re-used with different sets of input values. There's no need to make copies of the same notebook to edit and execute with new input values. EMR creates and saves the output notebook on S3 for each run of the parameterized notebook. For EMR notebook API code samples, see Sample commands to execute EMR Notebooks programmatically (p. 111).

> **Important**
> EMR Notebooks is supported with clusters created using Amazon EMR 5.18.0 and later. We strongly recommend that you use EMR Notebooks with clusters created using the latest version of Amazon EMR–particularly Amazon EMR release versions 5.30.0, 5.32.0 and later, or 6.2.0 and later. With these versions, a change was made so that Jupyter kernels run on the attached cluster, rather than on a Jupyter instance. This change helps improve performance and enhances your ability to customize kernels and libraries. For more information, see Differences in capabilities by cluster release version (p. 105).

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

Considerations when using EMR Notebooks

Consider the following requirements when you create clusters and develop solutions using EMR notebook.

Cluster requirements

- **Enable Amazon EMR Block Public Access** – Inbound access to a cluster enables cluster users to execute notebook kernels. Ensure that only authorized users can access the cluster. We strongly recommend that you leave block public access enabled, and that you limit inbound SSH traffic to only trusted sources. For more information, see Using Amazon EMR block public access (p. 542) and Control network traffic with security groups (p. 532).
- **Use a Compatible Cluster** – A cluster attached to a notebook must meet the following requirements:

- Only clusters created using Amazon EMR are supported. You can create a cluster independently within Amazon EMR and then attach an EMR notebook, or you can create a compatible cluster when you create an EMR notebook.
- Only clusters created using Amazon EMR release version 5.18.0 and later are supported. See the section called "Differences in capabilities by cluster release version" (p. 105).
- Clusters created using Amazon EC2 instances with AMD EPYC processors—for example, m5a.* and r5a.* instance types—are not supported.
- EMR Notebooks works only with clusters created with `VisibleToAllUsers` set to `true`. `VisibleToAllUsers` is `true` by default.
- 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 launched with Hadoop, Spark, and Livy installed. Other applications may be installed, but EMR Notebooks currently supports Spark clusters only.

  Important
  For EMR release versions 5.32.0 and later, or 6.2.0 and later, your cluster must also be running the Jupyter Enterprise Gateway application in order to work with EMR Notebooks.
- Clusters using Kerberos authentication are not supported.
- Clusters integrated with AWS Lake Formation support the installation of notebook-scoped libraries only. Installing kernels and libraries on the cluster are not supported.
- Clusters with multiple master nodes are not supported.
- Clusters using Amazon EC2 instances based on AWS Graviton2 are not supported.

Differences in capabilities by cluster release version

We strongly recommend that you use EMR Notebooks with clusters created using Amazon EMR release versions 5.30.0, 5.32.0 or later, or 6.2.0 or later. With these versions, EMR Notebooks runs kernels on the attached Amazon EMR cluster. Kernels and libraries can be installed directly on the cluster master node. Using EMR Notebooks with these cluster versions has the following benefits:

- **Improved performance** – Notebook kernels run on clusters with EC2 instance types that you select. Earlier versions run kernels on a specialized instance that cannot be resized, accessed, or customized.
- **Ability to add and customize kernels** – You can connect to the cluster to install kernel packages using `conda` and `pip`. In addition, `pip` installation is supported using terminal commands within notebook cells. In earlier versions, only pre-installed kernels were available (Python, PySpark, Spark, and SparkR). For more information, see Installing kernels and Python libraries on a cluster master node (p. 121).
- **Ability to install Python libraries** – You can install Python libraries on the cluster master node (p. 121) using `conda` and `pip`. We recommend using `conda`. With earlier versions, only notebook-scoped libraries (p. 123) for PySpark are supported.

Supported EMR Notebooks features by cluster release

Cluster release version	Notebook-scoped libraries for PySpark	Kernel installation on cluster	Python library installation on master node
Earlier than 5.18.0	EMR Notebooks not supported		
5.18.0–5.25.0	No	No	No
5.26.0–5.29.0	Yes (p. 123)	No	No
5.30.0	Yes (p. 123)	Yes (p. 121)	Yes (p. 121)

Cluster release version	Notebook-scoped libraries for PySpark	Kernel installation on cluster	Python library installation on master node
6.0.0	No	No	No
5.32.0 and later, and 6.2.0 and later	Yes (p. 123)	Yes (p. 121)	Yes (p. 121)

Limits for concurrently attached EMR Notebooks

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 EMR Notebooks
*.medium	2
*.large	4
*.xlarge	8
*.2xlarge	16
*.4xlarge	24
*.8xlarge	24
*.16xlarge	24

Jupyter Notebook and Python versions

EMR Notebooks runs Jupyter Notebook version 6.0.2 and Python 3.6.5 regardless of the Amazon EMR release version of the attached cluster.

If you specify an encrypted location in Amazon S3 to store notebook files, you must set up the Service role for EMR Notebooks (p. 409) as a key user. The default service role is `EMR_Notebooks_DefaultRole`. If you are using an AWS KMS key for encryption, see Using key policies in AWS KMS in the AWS Key Management Service Developer Guide and the support article for adding key users.

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 to which you want to attach the notebook, leave the default **Choose an existing cluster** selected, click **Choose**, select a cluster from the list, and then click **Choose cluster**. For information about cluster requirements for EMR Notebooks, see Considerations when using EMR Notebooks (p. 104).

   —or—

   Choose **Create a cluster**, enter a **Cluster name** and choose options according to the following guidelines. The cluster is created in the default VPC for the account using On-Demand instances.

Setting	Description
Cluster name	The friendly name used to identify the cluster.
Release	Cannot be modified. Defaults to the latest Amazon EMR release version (5.36.0).
Applications	Cannot be modified. Lists the applications that are installed on the cluster.
Instance	Enter the number of instances and select the EC2 Instance type. One instance is used for the master node. The rest are used for core nodes. The instance type determines the number of notebooks that can attach to the cluster simultaneously. For more information, see Limits for concurrently attached EMR Notebooks (p. 106).
EMR role	Leave the default or choose the link to specify a custom service role for Amazon EMR. For more information, see Service role for Amazon EMR (EMR role) (p. 398).
EC2 instance profile	Leave the default or choose the link to specify a custom service role for EC2 instances. For more information, see Service role for cluster EC2 instances (EC2 instance profile) (p. 403).
EC2 key pair	Choose an EC2 key pair to be able to connect to cluster instances. For more information, see Connect to the master node using SSH (p. 593).
Auto-termination	Auto-termination is supported for Amazon EMR versions 5.30.0 and 6.1.0 and later. Select the checkbox to enable auto-termination, then specify the amount of idle time after which the cluster should automatically shut down. For more information, see Using an auto-termination policy (p. 169).

5. For **Security groups**, choose **Use default security groups**. Alternatively, choose **Choose security groups** and select custom security groups that are available in the VPC of the cluster. You select one for the master instance and another for the notebook client instance. For more information, see the section called "Security groups for EMR Notebooks" (p. 541).
6. For **AWS Service Role**, leave the default or choose a custom role from the list. The client instance for the notebook uses this role. For more information, see Service role for EMR Notebooks (p. 409).

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`.

   If you specify an encrypted location in Amazon S3, you must set up the Service role for EMR Notebooks (p. 409) as a key user. The default service role is `EMR_Notebooks_DefaultRole`. If you are using an AWS KMS key for encryption, see Using key policies in AWS KMS in the AWS Key Management Service Developer Guide and the support article for adding key users.
8. Optionally, if you have added a Git-based repository to Amazon EMR that you want to associate with this notebook, choose **Git repository**, select **Choose repository** and then select a repository from the list. For more information, see Associating Git-based repositories with EMR Notebooks (p. 124).
9. 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 Use cluster and Notebook tags with IAM policies for access control (p. 387).
10. Choose **Create Notebook**.

Working with EMR 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

Status	Meaning
	cluster may still be provisioning resources or responding to other requests. You can open the notebook editor with the notebook in *local mode*. 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, `NotebookName.ipynb` remains in Amazon 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 notebook in Jupyter or JupyterLab 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 or JupyterLab. 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 can simply open the notebook editor within the Amazon EMR console.

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

> **Important**
> Amazon EMR creates a unique pre-signed URL for each notebook editor session, which is valid only for a short time. We recommend that you do not share the notebook editor URL. Doing this creates a security risk because recipients of the URL adopt your permissions to edit the notebook and run notebook code for the lifetime of the URL. If others need access to a notebook, provide permissions to their IAM user through permissions policies and ensure that the service role for EMR Notebooks has access to the Amazon S3 location. For more information, see the section called “Security” (p. 120) and Service role for EMR Notebooks (p. 409).

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 in JupyterLab** or **Open in Jupyter**.

   A new browser tab opens to the JupyterLab or 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 Cluster requirements (p. 104).
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 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.

The easiest way to open a notebook file from another user is to save the *.ipynb file from another user to your local file system, and then use the upload feature in the Jupyter and JupyterLab editors.

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-12A3BCDEFJHIJKLMNO45PQRST, 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 User Guide*.

```
aws s3 cp SharedNotebook.ipynb s3://MyBucket/
MyNotebooksFolder/-12A3BCDEFJHIJKLMNO45PQRST/MyNotebook.ipynb
```

Sample commands to execute EMR Notebooks programmatically

EMR Notebook execution APIs are available to execute EMR notebooks via a script or command line. The ability to start, stop, list, and describe EMR notebook executions without the AWS console enables you programmatically control an EMR notebook. Using a parameterized notebook cell, you can pass different

parameter values to a notebook without having to create a copy of the notebook for each new set of parameter values. See EMR API actions.

EMR notebook execution can be scheduled or batched using Amazon CloudWatch events and AWS Lambda. See Using AWS Lambda with Amazon CloudWatch Events

This section provides several examples of programmatic EMR notebook execution using the AWS CLI, Boto3 SDK (Python), and Ruby.

Notebook execution CLI command samples (p. 112)

Notebook execution Python samples (p. 115)

Notebook execution Ruby samples (p. 117)

You can also run parameterized notebooks as part of scheduled workflows using an orchestration tool such as Apache Airflow or Amazon Managed Workflows for Apache Airflow. For more information, see Orchestrating analytics jobs on EMR Notebooks using MWAA in the AWS Big Data Blog.

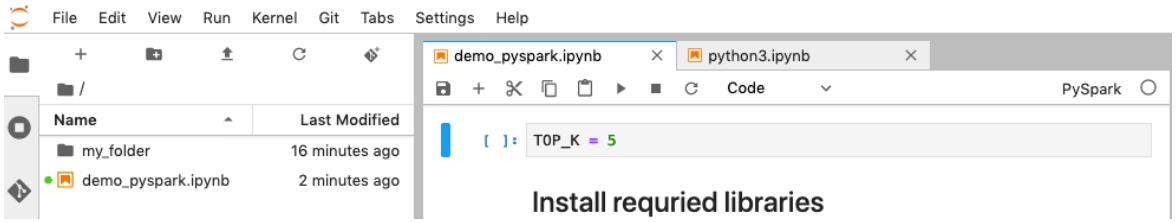
Limitations:

- A maximum of 100 concurrent executions are allowed per region per account.
- An execution is terminated if running for more than 30 days.

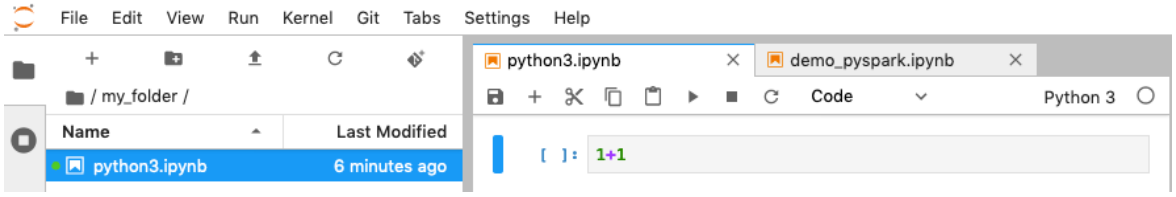
Notebook execution CLI command samples

The following is an example of the demo EMR Notebook seen from the EMR Notebook console. The notebook to run is located using the file path relative to the home directory. In this example, there are two notebook files that can be run: demo_pyspark.ipynb and my_folder/python3.ipynb.

The relative path for file `demo_pyspark.ipynb` is `demo_pyspark.ipynb`, shown below.



The relative path for `python3.ipynb` is `my_folder/python3.ipynb`, shown below.



For more information about the EMR API NotebookExecution actions, see EMR API actions.

Run a notebook

You can use the AWS CLI to run your notebook using the `start-notebook-execution` action, as the following example demonstrates.

```
aws emr --region us-east-1 \
start-notebook-execution \
```

```
--editor-id e-BKTM2DIHXBEDRU44ANWRKIU8N \
--notebook-params '{"input_param":"my-value", "good_superhero":["superman", "batman"]}' \
--relative-path test.ipynb \
--notebook-execution-name my-execution \
--execution-engine '{"Id" : "j-2QMOV6JAX1TS2"}' \
--service-role EMR_Notebooks_DefaultRole


{
    "NotebookExecutionId": "ex-IZWZZVR9DKQ9WQ7VZWXJZR29UGHTE"
}
```

Notebook output

Here's a sample notebook's output. Cell [3] shows the newly-injected parameter values.

Describe a notebook

You can use the `describe-notebook-execution` action to access information about a specific notebook execution.

```
aws emr --region us-east-1 \
describe-notebook-execution --notebook-execution-id ex-IZWZZVR9DKQ9WQ7VZWXJZR29UGHTE

{
    "NotebookExecution": {
        "NotebookExecutionId": "ex-IZWZZVR9DKQ9WQ7VZWXJZR29UGHTE",
        "EditorId": "e-BKTM2DIHXBEDRU44ANWRKIU8N",
        "ExecutionEngine": {
            "Id": "j-2QMOV6JAX1TS2",
            "Type": "EMR",
            "MasterInstanceSecurityGroupId": "sg-05ce12e58cd4f715e"
        },
        "NotebookExecutionName": "my-execution",
        "NotebookParams": "{\"input_param\":\"my-value\", \"good_superhero\":[\"superman\",
 \"batman\"]}",
        "Status": "FINISHED",
        "StartTime": 1593490857.009,
        "Arn": "arn:aws:elasticmapreduce:us-east-1:123456789012:notebook-execution/ex-
IZWZZVR9DKQ9WQ7VZWXJZR29UGHTE",
```

```
        "LastStateChangeReason": "Execution is finished for cluster j-2QMOV6JAX1TS2.",
        "NotebookInstanceSecurityGroupId": "sg-0683b0a39966d4a6a",
        "Tags": []
    }
}
```

Stop a notebook

If your notebook is running an execution that you'd like to stop, you can do so with the `stop-notebook-execution` command.

```
# stop a running execution
aws emr --region us-east-1 \
stop-notebook-execution --notebook-execution-id ex-IZWZX78UVPAATC8LHJR129B1RBN4T


# describe it
aws emr --region us-east-1 \
describe-notebook-execution --notebook-execution-id ex-IZWZX78UVPAATC8LHJR129B1RBN4T

{
    "NotebookExecution": {
        "NotebookExecutionId": "ex-IZWZX78UVPAATC8LHJR129B1RBN4T",
        "EditorId": "e-BKTM2DIHXBEDRU44ANWRKIU8N",
        "ExecutionEngine": {
            "Id": "j-2QMOV6JAX1TS2",
            "Type": "EMR"
        },
        "NotebookExecutionName": "my-execution",
        "NotebookParams": "{\"input_param\":\"my-value\", \"good_superhero\":[\"superman\",
 \"batman\"]}",
        "Status": "STOPPED",
        "StartTime": 1593490876.241,
        "Arn": "arn:aws:elasticmapreduce:us-east-1:123456789012:editor-execution/ex-
IZWZX78UVPAATC8LHJR129B1RBN4T",
        "LastStateChangeReason": "Execution is stopped for cluster j-2QMOV6JAX1TS2.
 Internal error",
        "Tags": []
    }
}
```

List a notebook's executions by start time

You can pass a `--from` parameter to `list-notebook-executions` to list your notebook's executions by start time.

```
# filter by start time
aws emr --region us-east-1 \
list-notebook-executions --from 1593400000.000

{
    "NotebookExecutions": [
        {
            "NotebookExecutionId": "ex-IZWZX78UVPAATC8LHJR129B1RBN4T",
            "EditorId": "e-BKTM2DIHXBEDRU44ANWRKIU8N",
            "NotebookExecutionName": "my-execution",
            "Status": "STOPPED",
            "StartTime": 1593490876.241
        },
        {
            "NotebookExecutionId": "ex-IZWZZVR9DKQ9WQ7VZWXJZR29UGHTE",
```

```
            "EditorId": "e-BKTM2DIHXBEDRU44ANWRKIU8N",
            "NotebookExecutionName": "my-execution",
            "Status": "RUNNING",
            "StartTime": 1593490857.009
        },
        {
            "NotebookExecutionId": "ex-IZWZYRS0M14L5V95WZ9OQ399SKMNW",
            "EditorId": "e-BKTM2DIHXBEDRU44ANWRKIU8N",
            "NotebookExecutionName": "my-execution",
            "Status": "STOPPED",
            "StartTime": 1593490292.995
        },
        {
            "NotebookExecutionId": "ex-IZX009ZK83IVY5E33VH8MDMELVK8K",
            "EditorId": "e-BKTM2DIHXBEDRU44ANWRKIU8N",
            "NotebookExecutionName": "my-execution",
            "Status": "FINISHED",
            "StartTime": 1593489834.765
        },
        {
            "NotebookExecutionId": "ex-IZWZXOZF88JWDF9J09GJ91R57VI0N",
            "EditorId": "e-BKTM2DIHXBEDRU44ANWRKIU8N",
            "NotebookExecutionName": "my-execution",
            "Status": "FAILED",
            "StartTime": 1593488934.688
        }
    ]
}
```

List a notebook's executions by start time and status

The `list-notebook-executions` command can also take a `--status` parameter to filter results.

```
# filter by start time and status
aws emr --region us-east-1 \
list-notebook-executions --from 1593400000.000 --status FINISHED
{
    "NotebookExecutions": [
        {
            "NotebookExecutionId": "ex-IZWZZVR9DKQ9WQ7VZWXJZR29UGHTE",
            "EditorId": "e-BKTM2DIHXBEDRU44ANWRKIU8N",
            "NotebookExecutionName": "my-execution",
            "Status": "FINISHED",
            "StartTime": 1593490857.009
        },
        {
            "NotebookExecutionId": "ex-IZX009ZK83IVY5E33VH8MDMELVK8K",
            "EditorId": "e-BKTM2DIHXBEDRU44ANWRKIU8N",
            "NotebookExecutionName": "my-execution",
            "Status": "FINISHED",
            "StartTime": 1593489834.765
        }
    ]
}
```

Notebook execution Python samples

The following code example is an SDK for Python (Boto3) file called `demo.py` that shows the notebook execution APIs.

For information about the EMR API NotebookExecution actions, see EMR API actions.

```
import boto3,time

emr = boto3.client(
    'emr',
    region_name='us-west-1'
)

start_resp = emr.start_notebook_execution(
    EditorId='e-40AC8ZO6EGGCPJ4DLO48KGGGI',
    RelativePath='boto3_demo.ipynb',
    ExecutionEngine={'Id':'j-1HYZS6JQKV11Q'},
    ServiceRole='EMR_Notebooks_DefaultRole'
)

execution_id = start_resp["NotebookExecutionId"]
print(execution_id)
print("\n")

describe_response = emr.describe_notebook_execution(NotebookExecutionId=execution_id)

print(describe_response)
print("\n")

list_response = emr.list_notebook_executions()
print("Existing notebook executions:\n")
for execution in list_response['NotebookExecutions']:
    print(execution)
    print("\n")

print("Sleeping for 5 sec...")
time.sleep(5)

print("Stop execution " + execution_id)
emr.stop_notebook_execution(NotebookExecutionId=execution_id)
describe_response = emr.describe_notebook_execution(NotebookExecutionId=execution_id)
print(describe_response)
print("\n")
```

Here's the output from running `demo.py`.

```
ex-IZX56YJDW1D29Q1PHR32WABU2SAPK

{'NotebookExecution': {'NotebookExecutionId': 'ex-IZX56YJDW1D29Q1PHR32WABU2SAPK',
 'EditorId': 'e-40AC8ZO6EGGCPJ4DLO48KGGGI', 'ExecutionEngine': {'Id': 'j-1HYZS6JQKV11Q',
 'Type': 'EMR'}, 'NotebookExecutionName': '', 'Status': 'STARTING', 'StartTime':
 datetime.datetime(2020, 8, 19, 0, 49, 19, 418000, tzinfo=tzlocal()), 'Arn':
 'arn:aws:elasticmapreduce:us-west-1:123456789012:notebook-execution/ex-
IZX56YJDW1D29Q1PHR32WABU2SAPK', 'LastStateChangeReason': 'Execution is starting for cluster
 j-1HYZS6JQKV11Q.', 'Tags': []}, 'ResponseMetadata': {'RequestId': '70f12c5f-1dda-45b7-
adf6-964987d373b7', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid':
 '70f12c5f-1dda-45b7-adf6-964987d373b7', 'content-type': 'application/x-amz-json-1.1',
 'content-length': '448', 'date': 'Wed, 19 Aug 2020 00:49:22 GMT'}, 'RetryAttempts': 0}}

Existing notebook executions:

{'NotebookExecutionId': 'ex-IZX56YJDW1D29Q1PHR32WABU2SAPK', 'EditorId':
 'e-40AC8ZO6EGGCPJ4DLO48KGGGI', 'NotebookExecutionName': '', 'Status': 'STARTING',
 'StartTime': datetime.datetime(2020, 8, 19, 0, 49, 19, 418000, tzinfo=tzlocal())}


{'NotebookExecutionId': 'ex-IZX5ABS5PR1E5AHMFYEMX3JJIORRB', 'EditorId':
 'e-40AC8ZO6EGGCPJ4DLO48KGGGI', 'NotebookExecutionName': '', 'Status': 'RUNNING',
 'StartTime': datetime.datetime(2020, 8, 19, 0, 48, 36, 373000, tzinfo=tzlocal())}
```

```
{'NotebookExecutionId': 'ex-IZX5GLVXIU1HNI8BWVW057F6MF4VE', 'EditorId':
 'e-40AC8ZO6EGGCPJ4DLO48KGGGI', 'NotebookExecutionName': '', 'Status': 'FINISHED',
 'StartTime': datetime.datetime(2020, 8, 19, 0, 45, 14, 646000, tzinfo=tzlocal()),
 'EndTime': datetime.datetime(2020, 8, 19, 0, 46, 26, 543000, tzinfo=tzlocal())}


{'NotebookExecutionId': 'ex-IZX5CV8YDUO8JAIWMXN2VH32RUIT1', 'EditorId':
 'e-40AC8ZO6EGGCPJ4DLO48KGGGI', 'NotebookExecutionName': '', 'Status': 'FINISHED',
 'StartTime': datetime.datetime(2020, 8, 19, 0, 43, 5, 807000, tzinfo=tzlocal()),
 'EndTime': datetime.datetime(2020, 8, 19, 0, 44, 31, 632000, tzinfo=tzlocal())}


{'NotebookExecutionId': 'ex-IZX5AS0PPW55CEDEURZ9NSOWSUJZ6', 'EditorId':
 'e-40AC8ZO6EGGCPJ4DLO48KGGGI', 'NotebookExecutionName': '', 'Status': 'FINISHED',
 'StartTime': datetime.datetime(2020, 8, 19, 0, 42, 29, 265000, tzinfo=tzlocal()),
 'EndTime': datetime.datetime(2020, 8, 19, 0, 43, 48, 320000, tzinfo=tzlocal())}


{'NotebookExecutionId': 'ex-IZX57YF5Q53BKWLR4I5QZ14HJ7DRS', 'EditorId':
 'e-40AC8ZO6EGGCPJ4DLO48KGGGI', 'NotebookExecutionName': '', 'Status': 'FINISHED',
 'StartTime': datetime.datetime(2020, 8, 19, 0, 38, 37, 81000, tzinfo=tzlocal()),
 'EndTime': datetime.datetime(2020, 8, 19, 0, 40, 39, 646000, tzinfo=tzlocal())}

Sleeping for 5 sec...
Stop execution ex-IZX56YJDW1D29Q1PHR32WABU2SAPK
{'NotebookExecution': {'NotebookExecutionId': 'ex-IZX56YJDW1D29Q1PHR32WABU2SAPK',
 'EditorId': 'e-40AC8ZO6EGGCPJ4DLO48KGGGI', 'ExecutionEngine': {'Id': 'j-1HYZS6JQKV11Q',
 'Type': 'EMR'}, 'NotebookExecutionName': '', 'Status': 'STOPPING', 'StartTime':
 datetime.datetime(2020, 8, 19, 0, 49, 19, 418000, tzinfo=tzlocal()), 'Arn':
 'arn:aws:elasticmapreduce:us-west-1:123456789012:notebook-execution/ex-
IZX56YJDW1D29Q1PHR32WABU2SAPK', 'LastStateChangeReason': 'Execution is being stopped for
 cluster j-1HYZS6JQKV11Q.', 'Tags': []}, 'ResponseMetadata': {'RequestId': '2a77ef73-
c1c6-467c-a1d1-7204ab2f6a53', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid':
 '2a77ef73-c1c6-467c-a1d1-7204ab2f6a53', 'content-type': 'application/x-amz-json-1.1',
 'content-length': '453', 'date': 'Wed, 19 Aug 2020 00:49:30 GMT'}, 'RetryAttempts': 0}}
```

Notebook execution Ruby samples

The following are Ruby code samples that demonstrate using the notebook execution API.

```
# prepare an EMR client

emr = Aws::EMR::Client.new(
  region: 'us-east-1',
  access_key_id: 'AKIA...JKPKA',
  secret_access_key: 'rLMeu...vU0OLrAC1',
)
```

Starting notebook execution and getting the execution id

In this example, the S3 editor and EMR notebook are `s3://mybucket/notebooks/e-EA8VGAA429FEQTC8HC9ZHWISK/test.ipynb`.

For information about the EMR API NotebookExecution actions, see EMR API actions.

```
start_response = emr.start_notebook_execution({
    editor_id: "e-EA8VGAA429FEQTC8HC9ZHWISK",
    relative_path: "test.ipynb",
```

```
    execution_engine: {id: "j-3U82I95AMALGE"},

    service_role: "EMR_Notebooks_DefaultRole",
})


notebook_execution_id = start_resp.notebook_execution_id
```

Describing notebook execution and printing the details

```
describe_resp = emr.describe_notebook_execution({
    notebook_execution_id: notebook_execution_id
})
puts describe_resp.notebook_execution
```

The output from the above commands will be as follows.

```
{
:notebook_execution_id=>"ex-IZX3VTVZWVWPP27KUB90BZ7V9IEDG",
:editor_id=>"e-EA8VGAA429FEQTC8HC9ZHWISK",
:execution_engine=>{:id=>"j-3U82I95AMALGE", :type=>"EMR", :master_instance_security_group_id=>nil},
:notebook_execution_name=>"",
:notebook_params=>nil,
:status=>"STARTING",
:start_time=>2020-07-23 15:07:07 -0700,
:end_time=>nil,
:arn=>"arn:aws:elasticmapreduce:us-east-1:123456789012:notebook-execution/ex-
IZX3VTVZWVWPP27KUB90BZ7V9IEDG",
:output_notebook_uri=>nil,
:last_state_change_reason=>"Execution is starting for cluster
 j-3U82I95AMALGE.", :notebook_instance_security_group_id=>nil,
:tags=>[]
}
```

Notebook filters

```


"EditorId": "e-XXXX",           [Optional]
"From" : "1593400000.000",    [Optional]
"To" :
```

Stopping notebook execution

```
stop_resp = emr.stop_notebook_execution({
    notebook_execution_id: notebook_execution_id
})
```

Enabling user impersonation to monitor 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 to 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. 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. 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:

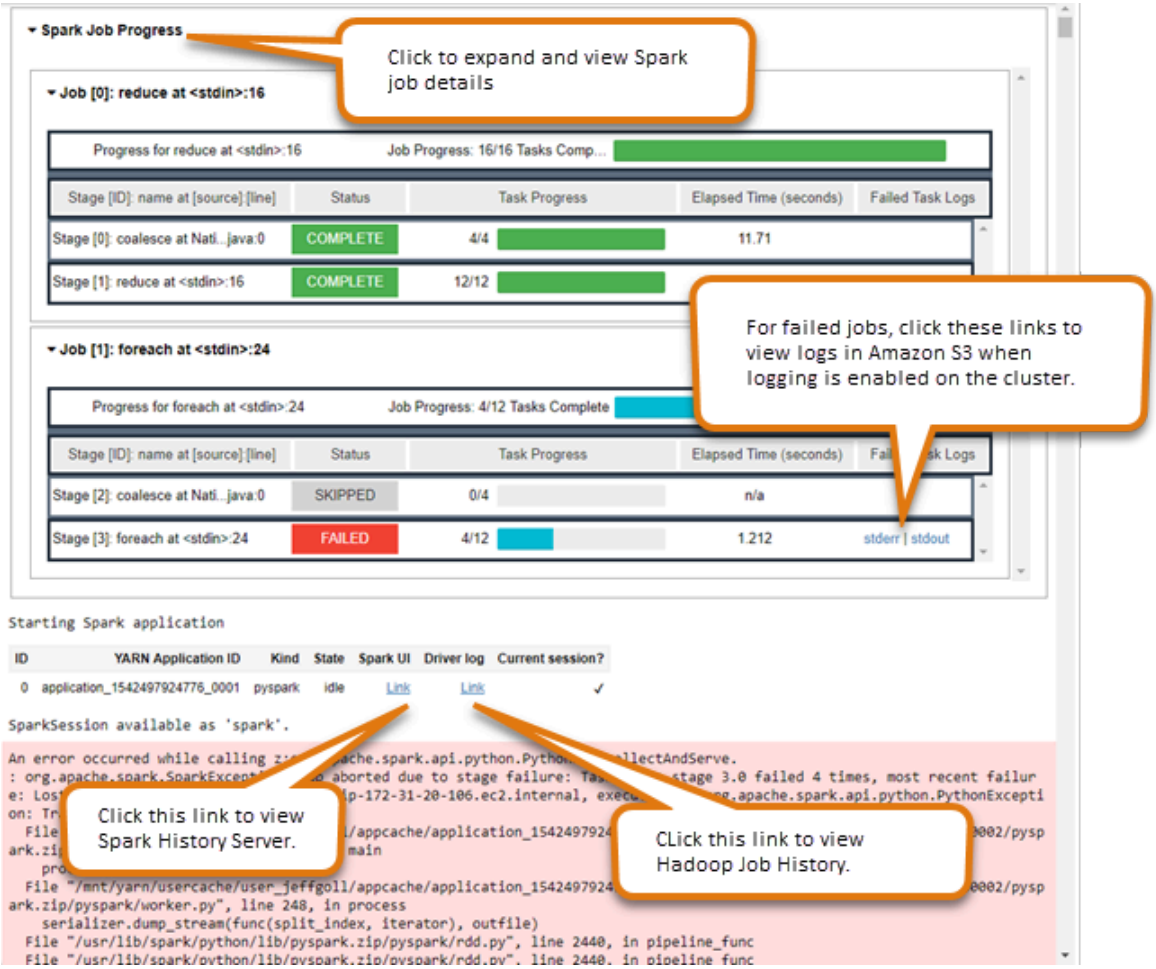
```
[
    {
        "Classification": "core-site",
        "Properties": {
          "hadoop.proxyuser.livy.groups": "*",
          "hadoop.proxyuser.livy.hosts": "*"
        }
    },
    {
        "Classification": "livy-conf",
        "Properties": {
          "livy.impersonation.enabled": "true"
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

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. 599). 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. 565).

The following is an example of the Spark job monitoring.



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 Amazon EMR (p. 353).

- You can use AWS Identity and Access Management policy statements together with notebook tags to limit access. For more information, see Condition keys (p. 387) and Example identity-based policy statements for EMR Notebooks (p. 441).
- 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 (p. 541).
- You specify an AWS Service Role that determines what permissions an EMR notebook has when interacting with other AWS services. For more information, see Service role for EMR Notebooks (p. 409).

Installing and using kernels and libraries

Each EMR notebook comes with a set of pre-installed libraries and kernels. You can install additional libraries and kernels in an Amazon EMR cluster if the cluster has access to the repository where the kernels and libraries are located. For example, for clusters in private subnets, you might need to configure network address translation (NAT) and provide a path for the cluster to access the public PyPI repository to install a library. For more information about configuring external access for different network configurations, see Scenarios and examples in the *Amazon VPC User Guide*.

Installing kernels and Python libraries on a cluster master node

With Amazon EMR release version 5.30.0 and later, excluding 6.0.0, you can install additional Python libraries and kernels on the master node of the cluster. After installation, these kernels and libraries are available to any user running an EMR notebook attached to the cluster. Python libraries installed this way are available only to processes running on the master node. The libraries are not installed on core or task nodes and are not available to executors running on those nodes.

> **Note**
> For EMR versions 5.30.1, 5.31.0, and 6.1.0, you must take additional steps in order to install kernels and libraries on the master node of a cluster.
> To enable the feature, do the following:

1. Make sure that the permissions policy attached to the service role for EMR Notebooks allows the following action:

   `elasticmapreduce:ListSteps`

   For more information, see Service role for EMR Notebooks.
2. Use the AWS CLI to run a step on the cluster that sets up EMR Notebooks as shown in the following example. You must use the step name `EMRNotebooksSetup`. Replace *`us-east-1`* with the Region in which your cluster resides. For more information, see Adding steps to a cluster using the AWS CLI.

```
aws emr add-steps --cluster-id MyClusterID --steps
 Type=CUSTOM_JAR,Name=EMRNotebooksSetup,ActionOnFailure=CONTINUE,Jar=s3://us-
east-1.elasticmapreduce/libs/script-runner/script-runner.jar,Args=["s3://
awssupportdatasvcs.com/bootstrap-actions/EMRNotebooksSetup/emr-notebooks-
setup.sh"]
```

You can install kernels and libraries using `pip` or `conda` in the `/emr/notebook-env/bin` directory on the master node.

Example – Installing Python libraries

From the Python3 kernel, run the `%pip` magic as a command from within a notebook cell to install Python libraries.

```
%pip install pmdarima
```

You may need to restart the kernel to use updated packages. You can also use the `%%sh` Spark magic to invoke `pip`.

```
%%sh
```

```
/emr/notebook-env/bin/pip install -U matplotlib
/emr/notebook-env/bin/pip install -U pmdarima
```

When using a PySpark kernel, you can either install libraries on the cluster using `pip` commands or use notebook-scoped libraries from within a PySpark notebook.

To run `pip` commands on the cluster from the terminal, first connect to the master node using SSH, as the following commands demonstrate.

```
sudo pip3 install -U matplotlib
sudo pip3 install -U pmdarima
```

Alternatively, you can use notebook-scoped libraries. With notebook-scoped libraries, your library installation is limited to the scope of your session and occurs on all Spark executors. For more information, see Using Notebook-scoped libraries.

If you want to package multiple Python libraries within a PySpark kernel, you can also create an isolated Python virtual environment. For examples, see Using Virtualenv.

To create a Python virtual environment in a session, use the Spark property `spark.yarn.dist.archives` from the `%%configure` magic command in the first cell in a notebook, as the following example demonstrates.

```
%%configure -f
{
   "conf": {
   "spark.yarn.appMasterEnv.PYSPARK_PYTHON":"./environment/bin/python",
   "spark.yarn.appMasterEnv.PYSPARK_DRIVER_PYTHON":"./environment/bin/python",
   "spark.yarn.dist.archives":"s3://DOC-EXAMPLE-BUCKET/prefix/
my_pyspark_venv.tar.gz#environment",
   "spark.submit.deployMode":"cluster"
   }
}
```

You can similarly create a Spark executor environment.

```
%%configure -f
{
   "conf": {
   "spark.yarn.appMasterEnv.PYSPARK_PYTHON":"./environment/bin/python",
   "spark.yarn.appMasterEnv.PYSPARK_DRIVER_PYTHON":"./environment/bin/python",
   "spark.executorEnv.PYSPARK_PYTHON":"./environment/bin/python",
   "spark.yarn.dist.archives":"s3://DOC-EXAMPLE-BUCKET/prefix/
my_pyspark_venv.tar.gz#environment",
   "spark.submit.deployMode":"cluster"
   }
}
```

You can also use `conda` to install Python libraries. Using `conda` requires sudo access. You must connect to the master node using SSH and then run `conda` from the terminal. For more information, see Connect to the master node using SSH (p. 593).

Example – Installing kernels

The following example demonstrates installing the Kotlin kernel using a terminal command while connected to the master node of a cluster:

```
sudo /emr/notebook-env/bin/conda install kotlin-jupyter-kernel -c jetbrains
```

Note
These instructions do not install kernel dependencies. If your kernel has third-party dependencies, you may need to take additional setup steps before you can use the kernel with your notebook.

Using Notebook-scoped libraries

Notebook-scoped libraries are available using clusters created using Amazon EMR release version 5.26.0 or later. Notebook-scoped libraries are intended to be used only with the PySpark kernel. Any user can install additional notebook-scoped libraries from within a notebook cell. These libraries are only available to that notebook user during a single notebook session. If other users need the same libraries, or the same user needs the same libraries in a different session, the library must be re-installed.

Considerations and limitations

Consider the following when using notebook-scoped libraries:

- You can uninstall only the libraries that are installed using `install_pypi_package` API. You cannot uninstall any libraries pre-installed on the cluster.
- If the same libraries with different versions are installed on the cluster and as notebook-scoped libraries, the notebook-scoped library version overrides the cluster library version.

Working with Notebook-scoped libraries

To install libraries, your Amazon EMR cluster must have access to the PyPI repository where the libraries are located.

The following examples demonstrate simple commands to list, install, and uninstall libraries from within a notebook cell using the PySpark kernel and APIs. For additional examples, see Install Python libraries on a running cluster with EMR Notebooks post on the AWS Big Data Blog.

Example – Listing current libraries

The following command lists the Python packages available for the current Spark notebook session. This lists libraries installed on the cluster and notebook-scoped libraries.

```
sc.list_packages()
```

Example – Installing the Celery library

The following command installs the Celery library as a notebook-scoped library.

```
sc.install_pypi_package("celery")
```

After installing the library, the following command confirms that the library is available on the Spark driver and executors.

```
import celery
sc.range(1,10000,1,100).map(lambda x: celery.__version__).collect()
```

Example – Installing the Arrow library, specifying the version and repository

The following command installs the Arrow library as a notebook-scoped library, with a specification of the library version and repository URL.

```
sc.install_pypi_package("arrow==0.14.0", "https://pypi.org/simple")
```

Example – Uninstalling a library

The following command uninstalls the Arrow library, removing it as a notebook-scoped library from the current session.

```
sc.uninstall_package("arrow")
```

Associating Git-based repositories with EMR Notebooks

You can associate Git-based repositories with your Amazon EMR notebooks to save your notebooks in a version controlled environment. You can associate up to three repositories with a notebook. The following Git-based services are supported:

- AWS CodeCommit
- GitHub
- Bitbucket
- GitLab

Associating Git-based repositories with your notebook has the following benefits.

- **Version control** – You can record code changes in a version-control system so that you can review the history of your changes and selectively reverse them.
- **Collaboration** – Colleagues working in different notebooks can share code through remote Git-based repositories. Notebooks can clone or merge code from remote repositories and push changes back to those remote repositories.
- **Code reuse** – Many Jupyter notebooks that demonstrate data analysis or machine learning techniques are available in publicly hosted repositories, such as GitHub. You can associate your notebooks with a repository to reuse the Jupyter notebooks contained in a repository.

To use Git-based repositories with EMR Notebooks, you add the repositories as resources in the Amazon EMR console, associate credentials for repositories that require authentication, and link them with your notebooks. You can view a list of repositories that are stored in your account and details about each repository in the Amazon EMR console. You can associate an existing Git-based repository with a notebook when you create it.

Topics

- Prerequisites and considerations (p. 125)
- Add a Git-based repository to Amazon EMR (p. 126)
- Update or delete a Git-based repository (p. 128)
- Link or unlink a Git-based repository (p. 128)

- Create a new Notebook with an associated Git repository (p. 129)
- Use Git repositories in a Notebook (p. 130)

Prerequisites and considerations

Consider the following when planning to integrate a Git-based repository with EMR Notebooks.

AWS CodeCommit

If you use a CodeCommit repository, you must use Git credentials and HTTPS with CodeCommit. SSH Keys, and HTTPS with the AWS CLI credential helper are not supported. CodeCommit does not support personal access tokens (PATs). For more information, see Using IAM with CodeCommit: Git credentials, SSH keys, and AWS access keys in the *IAM User Guide* and Setup for HTTPS users using Git credentials in the *AWS CodeCommit User Guide*.

Access and permission considerations

Before associating a repository with your notebook, make sure that your cluster, IAM role for EMR Notebooks, and security groups have the correct settings and permissions. You can also configure Git-based repositories that you host in a private network by following the instructions in Configure a privately-hosted Git repository for EMR Notebooks (p. 125).

- **Cluster internet access** – The network interface that is launched has only a private IP address. This means that the cluster that your notebook connects to must be in a private subnet with a network address translation (NAT) gateway or must be able to access the internet through a virtual private gateway. For more information, see Amazon VPC options.

  The security groups for your notebook must include an outbound rule that allows the notebook to route traffic to the internet from the cluster. We recommend that you create your own security groups. For more information, see Specifying EC2 security groups for EMR Notebooks.

  > **Important**
  > If the network interface is launched into a public subnet, it won't be able to communicate with the internet through an internet gateway (IGW).

- **Permissions for AWS Secrets Manager** – If you use Secrets Manager to store secrets that you use to access a repository, the the section called "EMR Notebooks role" (p. 409) must have a permissions policy attached that allows the `secretsmanager:GetSecretValue` action.

Configure a privately-hosted Git repository for EMR Notebooks

Use the following instructions to configure privately-hosted repositories for EMR Notebooks. You must provide a configuration file with information about your DNS and Git servers. Amazon EMR uses this information to configure EMR notebooks that can route traffic to your privately-hosted repositories.

Prerequisites

Before you configure a privately-hosted Git repository for EMR Notebooks, you must have the following:

- An Amazon S3 Control location where files for your EMR notebook will be saved.

To configure one or more privately-hosted Git repositories for EMR Notebooks

1. Create a configuration file using the provided template. Include the following values for each Git server that you want to specify in your configuration:

- **`DnsServerIpV4`**- The IPv4 address of your DNS server. If you provide values for both `DnsServerIpV4` and `GitServerIpV4List`, the value for `DnsServerIpV4` takes precedence and will be used to resolve your `GitServerDnsName`.

  Note
  To use privately-hosted Git repositories, your DNS server must allow inbound access from EMR Notebooks. We strongly recommend that you secure your DNS server against other, unauthorized access.
- **`GitServerDnsName`** - The DNS name of your Git server. For example `"git.example.com"`.
- **`GitServerIpV4List`** - A list of IPv4 addresses that belong to your Git server(s).

```
[
    {
        "Type": "PrivatelyHostedGitConfig",
        "Value": [
            {
                "DnsServerIpV4": "<10.24.34.xxx>",
                "GitServerDnsName": "<enterprise.git.com>",
                "GitServerIpV4List": [
                    "<xxx.xxx.xxx.xxx>",
                    "<xxx.xxx.xxx.xxx>"
                ]
            },
            {
                "DnsServerIpV4": "<10.24.34.xxx>",
                "GitServerDnsName": "<git.example.com>",
                "GitServerIpV4List": [
                    "<xxx.xxx.xxx.xxx>",
                    "<xxx.xxx.xxx.xxx>"
                ]
            }
        ]
    }
]
```

2. Save your configuration file as `configuration.json`.
3. Upload the configuration file into your designated Amazon S3 storage location in a folder called `life-cycle-configuration`. For example, if your default S3 location is `s3://DOC-EXAMPLE-BUCKET/notebooks`, your configuration file should be located at `s3://DOC-EXAMPLE-BUCKET/notebooks/`**`life-cycle-configuration/configuration.json`**.

   Important
   We strongly recommend that you restrict access to your `life-cycle-configuration` folder to only your EMR Notebooks administrators, and to the service role for EMR Notebooks. You should also secure `configuration.json` against unauthorized access. For instructions, see Controlling access to a bucket with user policies or Security Best Practices for Amazon S3.

   For upload instructions, see Creating a folder and Uploading objects in the *Amazon Simple Storage Service User Guide*.

Add a Git-based repository to Amazon EMR

To add a Git-based repository as a resource in your Amazon EMR account

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose **Git repositories**, and then choose **Add repository**.

3. For **Repository name**, enter a name to use for the repository in Amazon EMR.

   Names may only contain alphanumeric characters, hyphens (-), or underscores (_).
4. For **Git repository URL**, enter the URL for the repository. When using a CodeCommit repository, this is the URL that is copied when you choose **Clone URL** and then **Clone HTTPS**, for example, `https://git-codecommit.us-west-2.amazonaws.com/v1/repos/MyCodeCommitRepoName`.
5. For **Branch**, enter a branch name.
6. For **Git credentials**, choose options according to the following guidelines. You can use a Git user name and password or a personal access token (PAT) to authenticate to your repository. EMR Notebooks accesses your Git credentials using secrets stored in Secrets Manager.

   Note
   If you use a GitHub repository, we recommend that you use a personal access token (PAT) to authenticate. Beginning August 13, 2021, GitHub will no longer accept passwords when authenticating Git operations. For more information, see the Token authentication requirements for Git operations post in *The GitHub Blog*.

Option	Description
Use an existing AWS secret	Choose this option if you already saved your credentials as a secret in Secrets Manager, and then select the secret name from the list. If you select a secret associated with a Git user name and password, the secret must be in the format `{"gitUsername": "MyUserName", "gitPassword": "MyPassword"}`.
Create a new secret	Choose this option to associate existing Git credentials with a new secret that you create in Secrets Manager. Do one of the following based on the Git credentials that you use for the repository. If you use a Git user name and password to access the repository, select **Username and password**, enter the **Secret name** to use in Secrets Manager, and then enter the **Username** and **Password** to associate with the secret. –OR– If you use a personal access token to access the repository, select **Personal access token (PAT)**, enter the **Secret name** to use in Secrets Manager, and then enter your **personal access token**. For more information, see Creating a personal access token for the command line for GitHub and Personal access tokens for Bitbucket. CodeCommit repositories do not support this option.
Use a public repository without credentials	Choose this option to access a public repository.

7. Choose **Add repository**.

Update or delete a Git-based repository

To update a Git-based repository

1. On the **Git repositories** page, choose the repository you want to update.
2. On the repository page, choose **Edit repository**.
3. Update **Git credentials** on the repository page.

To delete a Git repository

1. On the **Git repositories** page, choose the repository you want to delete.
2. On the repository page, choose all the notebooks that are currently linked to the repository. Choose **Unlink notebook**.
3. On the repository page, choose **Delete**.

> **Note**
> To delete the local Git repository from Amazon EMR, you must first unlink any notebooks from this repository. For more information, see Link or unlink a Git-based repository (p. 128). Deleting a Git repository will not delete any secret created for the repository. You can delete the secret in AWS Secrets Manager.

Link or unlink a Git-based repository

To link a Git-based repository to an EMR notebook

The repository can be linked to a notebook once the notebook is **Ready**.

1. From the **Notebooks** list, choose the notebook you want to update.
2. In the **Git repositories** section on the **Notebook** page, choose **Link new repository**.
3. In the repository list of the **Link Git repository to notebook** window, select one or more repositories that you want to link to your notebook, and then choose **Link repository**.

Or

1. On the **Git repositories** page, choose the repository you want to link to your notebook.
2. In the list of **EMR notebooks**, choose **Link new notebook** to link this repository to an existing notebook.

To unlink a Git repository from an EMR notebook

1. From the **Notebooks** list, choose the notebook you want to update.
2. In the list of **Git repositories**, select the repository that you want to unlink from your notebook, and then choose **Unlink repository**.

Or

1. On the **Git repositories** page, choose the repository you want to make updates to.
2. In the list of **EMR notebooks**, select the notebook that you want to unlink from the repository, and then choose **Unlink notebook**.

Note
Linking a Git repository to a notebook clones the remote repository to your local Jupyter notebook. Unlinking the Git repository from a notebook only disconnects the notebook from the remote repository but does not delete the local Git repository.

Understanding repository status

A Git repository may have any of the following status in the repository list. For more information about linking EMR notebooks with Git repositories, see Link or unlink a Git-based repository (p. 128).

Status	Meaning
Linking	The Git repository is being linked to the notebook. While the repository is **Linking**, you cannot stop the notebook.
Linked	The Git repository is linked to the notebook. While the repository has a **Linked** status, it is connected to the remote repository.
Link Failed	The Git repository failed to link to the notebook. You can try again to link it.
Unlinking	The Git repository is being unlinked from the notebook. While the repository is **Unlinking**, you cannot stop the notebook. Unlinking a Git repository from a notebook only disconnects it from the remote repository; it doesn't delete any code from the notebook.
Unlink Failed	The Git repository failed to unlink from the notebook. You can try again to unlink it.

Create a new Notebook with an associated Git repository

To create a notebook and associate it with Git repositories in the AWS Management Console

1. Follow the instructions at Creating a Notebook (p. 106).
2. For **Security group**, choose **Use your own security group**.

   Note
   The security groups for your notebook must include an outbound rule to allow the notebook to route traffic to the internet via the cluster. It is recommended that you create your own security groups. For more information, see Specifying EC2 security groups for EMR Notebooks.
3. For **Git repositories**, **Choose repository** to associate with the notebook.

   1. Choose a repository that is stored as a resource in your account, and then choose **Save**.
   2. To add a new repository as a resource in your account, choose **add a new repository**. Complete the **Add repository** workflow in a new window.

Use Git repositories in a Notebook

You can choose to **Open in JupyterLab** or **Open in Jupyter** when you open a notebook.

If you choose to open the notebook in Jupyter, a list of expandable files and folders within the notebook are displayed. You can manually run Git commands like the following in a notebook cell.

```
!git pull origin master
```

To open any of the additional repositories, navigate to other folders.

If you choose to open the notebook with a JupyterLab interface, the jupyter-git extension is installed and available to use. For information about the jupyter-git extension for JupyterLab, see jupyterlab-git.

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. 131).
- Whether you are running Amazon EMR clusters on Outposts or Local Zones. See EMR clusters on AWS Outposts (p. 160) or EMR clusters on AWS Local Zones (p. 162).
- Whether a cluster is long-running or transient, and what software it runs. See Configuring a cluster to continue or terminate after step execution (p. 167) and Configure cluster software (p. 190).
- Whether a cluster has a single primary node or three primary nodes. See Plan and configure master nodes (p. 148).
- The hardware and networking options that optimize cost, performance, and availability for your application. See Configure cluster hardware and networking (p. 195).
- 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. 342) and Tag clusters (p. 348).
- How to authenticate and authorize access to cluster resources, and how to encrypt data. See Security in Amazon EMR (p. 353).
- How to integrate with other software and services. See Drivers and third-party application integration (p. 352).

Launch a cluster quickly

Use the **Create Cluster - Quick Options** page in the Amazon EMR console to quickly create a cluster for simple tasks or for evaluation or testing purposes. **Quick Options** uses default values for configuration options like cluster software, networking, and security.

To launch a cluster with Quick Options with the console

1. Sign in to the AWS Management Console and open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose **Clusters**, and then choose **Create cluster** to open the **Quick Options** page.
3. On the **Create Cluster - Quick Options** page, enter or select values for the provided fields.
4. Choose **Create cluster** to launch the cluster and open the cluster status page.
5. On the cluster status page, find the cluster **Status** next to the cluster name. The status should change from **Starting** to **Running** to **Waiting** during the cluster creation process. You may need to choose the refresh icon on the right or refresh your browser to receive updates.

   When the status changes to **Waiting**, your cluster is up, running, and ready to accept steps and SSH connections.

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. 132)
- Work with storage and file systems (p. 133)
- Prepare input data (p. 135)
- Configure an output location (p. 144)

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://DOC-EXAMPLE-BUCKET1/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.

> **Important**
> Beginning with Amazon EMR release 5.22.0, Amazon EMR uses AWS Signature Version 4 exclusively to authenticate requests to Amazon S3. Earlier Amazon EMR releases use AWS Signature Version 2 in some cases, unless the release notes indicate that Signature Version 4 is used exclusively. For more information, see Authenticating Requests (AWS Signature Version 4) and Authenticating Requests (AWS Signature Version 2) in the *Amazon Simple Storage Service Developer Guide*.

File system	Prefix	Description
HDFS	`hdfs://` (or no prefix)	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. 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.
EMRFS	`s3://`	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. **Note** Previously, Amazon EMR used the `s3n` and `s3a` file systems. While both still work, we

File system	Prefix	Description
		recommend that you use the `s3` URI scheme for the best performance, security, and reliability.
local file system		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 *instance store*. 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. The local file system is used by HDFS, but Python also runs from the local file system and you can choose to store additional application files on instance store volumes.
(Legacy) Amazon S3 block file system	`s3bfs://`	The Amazon S3 block file system is a legacy file storage system. We strongly discourage the use of this system. **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.

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. 135)
- How to get data into Amazon EMR (p. 136)

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 DistributedCache 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. 136)
- Upload data with AWS DataSync (p. 140)
- Import files with distributed cache (p. 140)
- How to process compressed files (p. 143)
- Import DynamoDB data into Hive (p. 143)
- Connect to data with AWS Direct Connect (p. 143)
- Upload large amounts of data with AWS Snowball (p. 144)

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 User 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. 136)
- Configure multipart upload for Amazon S3 (p. 137)
- Best practices (p. 139)

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 User 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 S3 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 S3 User Guide*.
- To create a bucket using an SDK, see Examples of creating a bucket in the *Amazon Simple Storage Service User 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 User Guide*.

- To work with buckets using Amazon S3 Access Points, see Using a bucket-style alias for your access point in the *Amazon S3 User Guide*. You can easily use Amazon S3 Access Points with the Amazon S3 Access Point Alias instead of the Amazon S3 bucket name. You can use the Amazon S3 Access Point Alias for both existing and new applications, including Spark, Hive, Presto and others.

  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 User 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. For more information, see Configure applications in the *Amazon EMR Release Guide*.

Configuration parameter name	Default value	Description
`fs.s3n.multipart.uploads.enabled`	`true`	A Boolean type that indicates whether to enable multipart uploads. When EMRFS consistent view is enabled, multipart uploads are enabled by default and setting this value to `false` is ignored.
`fs.s3n.multipart.uploads.split.size`	`134217728`	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 `PutObject` request to upload the file. For more information, see

Configuration parameter name	Default value	Description
`fs.s3n.ssl.enabled`	`true`	A Boolean type that indicates whether to use http or https.
`fs.s3.buckets.create.enabled`	`false`	A Boolean type that indicates whether a bucket should be created if it does not exist. Setting to `false` causes an exception on `CreateBucket` operations.
`fs.s3.multipart.clean.enabled`	`false`	A Boolean type that indicates whether to enable background periodic clean-up of incomplete multipart uploads.
`fs.s3.multipart.clean.age.threshold`	`604800`	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.
`fs.s3.multipart.clean.jitter.max`	`10000`	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/emr.
2. Choose **Create cluster**, **Go to advanced options**.
3. Under **Edit Software Settings** and enter the following configuration: `classification=core-site,properties=[fs.s3n.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-5.36.0 --applications Name=Hive Name=Pig \
--use-default-roles --ec2-attributes KeyName=myKey --instance-type m5.xlarge \
--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 User 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 User 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. 137). 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 `DaysAfterIntitiation` 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. 137).

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 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 User Guide*.

Upload data with AWS DataSync

AWS DataSync is an online data transfer service that simplifies, automates, and accelerates the process of moving data between your on-premises storage and AWS storage services or between AWS storage services. DataSync supports a variety of on-premises storage systems such as Hadoop Distributed File System (HDFS), NAS file servers, and self-managed object storage.

The most common way to get data onto a cluster is to upload the data to Amazon S3 and use the built-in features of Amazon EMR to load the data onto your cluster.

DataSync can help you accomplish the following tasks:

- Replicate HDFS on your Hadoop cluster to Amazon S3 for business continuity
- Copy HDFS to Amazon S3 to populate your data lakes
- Transfer data between your Hadoop cluster's HDFS and Amazon S3 for analysis and processing

To upload data to your S3 bucket, you first deploy one or more DataSync agents in the same network as your on-premises storage. An *agent* is a virtual machine (VM) that is used to read data from or write data to a self-managed location. You then activate your agents in the AWS account and AWS Region where your S3 bucket is located.

After your agent is activated, you create a source location for your on-premises storage, a destination location for your S3 bucket, and a task. A *task* is a set of two locations (source and destination) and a set of default options that you use to control the behavior of the task.

Finally, you run your DataSync task to transfer data from the source to the destination.

For more information, see Getting started with AWS DataSync.

Import files with distributed cache

Topics

- Supported file types (p. 141)
- Location of cached files (p. 141)
- Access cached files from streaming applications (p. 141)
- Access cached files from streaming applications using the Amazon EMR console (p. 141)
- Access cached files from streaming applications using the AWS CLI (p. 142)

DistributedCache 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 DistributedCache 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 DistributedCache 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. 191).

Supported file types

DistributedCache 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`. DistributedCache passes the compressed files to each core node and decompresses the archive as part of caching. DistributedCache supports the following compression formats:

- zip
- tgz
- tar.gz
- tar
- jar

Location of cached files

DistributedCache copies files to core nodes only. If there are no core nodes in the cluster, DistributedCache copies the files to the master node.

DistributedCache 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, DistributedCache puts the file in the `archive` directory. If you cache an archive, DistributedCache 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 DistributedCache 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/emr.
2. Choose **Create cluster**.
3. Choose **Step execution** as the Launch mode.

4. In the **Steps** section, in the **Add step** field, choose **Streaming program** from the list and click **Configure and add**.
5. In the **Arguments** field, include the files and archives to save to the cache and click **Add**.

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

If you want to ...	Action	Example	
Add an individual file to the distributed cache	Specify `-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.	`-cacheFile \ 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 m5.xlarge --instance-count 3 --steps Type=STREAMING,Name="Streaming
 program",ActionOnFailure=CONTINUE,Args=["--files","s3://my_bucket/my_mapper.py s3://
my_bucket/my_reducer.py","-mapper","my_mapper.py","-reducer","my_reducer.py,"-input","s3://
my_bucket/my_input","-output","s3://my_bucket/my_output", "-cacheFile","s3://my_bucket/
sample_dataset.dat#sample_dataset_cached.dat"]
```

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

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

Example 2

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

```
aws emr create-cluster --name "Test cluster" --release-label emr-4.0.0 --
applications Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey
 --instance-type m5.xlarge --instance-count 3 --steps Type=STREAMING,Name="Streaming
 program",ActionOnFailure=CONTINUE,Args=["--files","s3://my_bucket/my_mapper.py s3://
my_bucket/my_reducer.py","-mapper","my_mapper.py","-reducer","my_reducer.py,"-input","s3://
my_bucket/my_input","-output","s3://my_bucket/my_output", "-cacheArchive","s3://my_bucket/
sample_dataset.tgz#sample_dataset_cached"]
```

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

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

How to process compressed files

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

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

Import DynamoDB data into Hive

The implementation of Hive provided by Amazon EMR includes functionality that you can use to import and export data between DynamoDB and an Amazon EMR cluster. This is useful if your input data is stored in DynamoDB. For more information, see Export, import, query, and join tables in DynamoDB using Amazon EMR.

Connect to data with AWS Direct Connect

AWS Direct Connect is a service you can use to establish a private dedicated network connection to Amazon Web Services from your data center, 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 Snowball

AWS Snowball is a service you can use to transfer large amounts of data between Amazon Simple Storage Service (Amazon S3) and your onsite data storage location at faster-than-internet speeds. Snowball supports two job types: import jobs and export jobs. Import jobs involve a data transfer from an on-premises source to an Amazon S3 bucket. Export jobs involve a data transfer from an Amazon S3 bucket to an on-premises source. For both job types, Snowball devices secure and protect your data while regional shipping carriers transport them between Amazon S3 and your onsite data storage location. Snowball devices are physically rugged and protected by the AWS Key Management Service (AWS KMS). For more information, see the AWS Snowball Edge 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. 144)
- What formats can Amazon EMR return? (p. 145)
- How to write data to an Amazon S3 bucket you don't own (p. 145)
- Compress the output of your cluster (p. 147)

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*.

To create a an Amazon S3 bucket, follow the instructions on the Creating a bucket page in the *Amazon Simple Storage Service Developers Guide*.

> **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. We strongly recommend that you follow Security Best Practices for Amazon S3 when configuring your bucket.

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://DOC-EXAMPLE-BUCKET1/script/MapperScript.py`
log files	`s3://DOC-EXAMPLE-BUCKET1/logs`
input data	`s3://DOC-EXAMPLE-BUCKET1/input`
output data	`s3://DOC-EXAMPLE-BUCKET1/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 User Guide*.

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

Canned ACL	Description
`AuthenticatedRead`	Specifies that the owner is granted `Permission.FullControl` and the `GroupGrantee.AuthenticatedUsers` group grantee is granted `Permission.Read` access.

Canned ACL	Description
`BucketOwnerFullControl`	Specifies that the owner of the bucket is granted `Permission.FullControl`. The owner of the bucket is not necessarily the same as the owner of the object.
`BucketOwnerRead`	Specifies that the owner of the bucket is granted `Permission.Read`. The owner of the bucket is not necessarily the same as the owner of the object.
`LogDeliveryWrite`	Specifies that the owner is granted `Permission.FullControl` and the `GroupGrantee.LogDelivery` group grantee is granted `Permission.Write` access, so that access logs can be delivered.
`Private`	Specifies that the owner is granted `Permission.FullControl`.
`PublicRead`	Specifies that the owner is granted `Permission.FullControl` and the `GroupGrantee.AllUsers` group grantee is granted `Permission.Read` access.
`PublicReadWrite`	Specifies that the owner is granted `Permission.FullControl` and the `GroupGrantee.AllUsers` group grantee is granted `Permission.Read` and `Permission.Write` 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. 593).

  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(*) 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. 593).

  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. 147)
- Intermediate data compression (p. 148)
- Using the Snappy library with Amazon EMR (p. 148)

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/.

Plan and configure master nodes

When you launch an Amazon EMR cluster, you can choose to have one or three master nodes in your cluster. Launching a cluster with three master nodes is only supported by Amazon EMR version 5.23.0 and later. Amazon EMR can take advantage of EC2 placement groups to ensure master nodes are placed on distinct underlying hardware to further improve cluster availability. For more information, see Amazon EMR integration with EC2 placement groups (p. 156).

An EMR cluster with multiple master nodes provides the following key benefits:

- The master node is no longer a single point of failure. If one of the master nodes fails, the cluster uses the other two master nodes and runs without interruption. In the meantime, Amazon EMR automatically replaces the failed master node with a new one that is provisioned with the same configuration and bootstrap actions.
- Amazon EMR enables the Hadoop high availability features of HDFS NameNode and YARN ResourceManager and supports high availability for a few other open source applications.

  For more information about how an EMR cluster with multiple master nodes supports open source applications and other Amazon EMR features, see Supported applications and features (p. 149).

  Note
  The cluster can reside only in one Availability Zone or subnet.

This section provides information about supported applications and features of an EMR cluster with multiple master nodes as well as the configuration details, best practices, and considerations for launching the cluster.

Topics

- Supported applications and features (p. 149)
- Launch an Amazon EMR Cluster with multiple master nodes (p. 154)
- Amazon EMR integration with EC2 placement groups (p. 156)
- Considerations and best practices (p. 160)

Supported applications and features

This topic provides information about the Hadoop high availability features of HDFS NameNode and YARN ResourceManager in an Amazon EMR cluster, and how the high availability features work with open source applications and other Amazon EMR features.

High availability HDFS

An EMR cluster with multiple master nodes enables the HDFS NameNode high availability feature in Hadoop. For more information, see HDFS high availability.

In an Amazon EMR cluster, two or more separate nodes are configured as NameNodes. One NameNode is in an `active` state and the others are in a `standby` state. If the node with `active` NameNode fails, Amazon EMR starts an automatic HDFS failover process. A node with `standby` NameNode becomes `active` and takes over all client operations in the cluster. Amazon EMR replaces the failed node with a new one, which then rejoins as a `standby`.

> **Note**
> In Amazon EMR versions 5.23.0 up to and including 5.30.1, only two of the three master nodes run HDFS NameNode.

If you need to find out which NameNode is `active`, you can use SSH to connect to any master node in the cluster and run the following command:

```
hdfs haadmin -getAllServiceState
```

The output lists the nodes where NameNode is installed and their status. For example,

```
ip-##-#-#-##1.ec2.internal:8020 active
ip-##-#-#-##2.ec2.internal:8020 standby
ip-##-#-#-##3.ec2.internal:8020 standby
```

High availability YARN ResourceManager

An EMR cluster with multiple master nodes enables the YARN ResourceManager high availability feature in Hadoop. For more information, see ResourceManager high availability.

In an EMR cluster with multiple master nodes, YARN ResourceManager runs on all three master nodes. One ResourceManager is in `active` state, and the other two are in `standby` state. If the master node with `active` ResourceManager fails, Amazon EMR starts an automatic failover process. A master node with a `standby` ResourceManager takes over all operations. Amazon EMR replaces the failed master node with a new one, which then rejoins the ResourceManager quorum as a `standby`.

You can connect to "http://*`master-public-dns-name`*:8088/cluster" for any master node, which automatically directs you to the `active` resource manager. To find out which resource manager is `active`, use SSH to connect to any master node in the cluster. Then run the following command to get a list of the three master nodes and their status:

```
yarn rmadmin -getAllServiceState
```

Supported applications in an Amazon EMR Cluster with multiple master nodes

You can install and run the following applications on an EMR cluster with multiple master nodes. For each application, the master node failover process varies.

Application	Availability during master node failover	Notes
Flink	Availability not affected by master node failover	Flink jobs on Amazon EMR run as YARN applications. Flink's JobManagers run as YARN's ApplicationMasters on core nodes. The JobManager is not affected by the master node failover process. If you use Amazon EMR version 5.27.0 or earlier, the JobManager is a single point of failure. When the JobManager fails, it loses all job states and will not resume the running jobs. You can enable JobManager high availability by configuring application attempt count, checkpointing, and enabling ZooKeeper as state storage for Flink. For more information, see Configuring Flink on an Amazon EMR Cluster with multiple master nodes. Beginning with Amazon EMR version 5.28.0, no manual configuration is needed to enable JobManager high availability.
Ganglia	Availability not affected by master node failover	Ganglia is available on all master nodes, so Ganglia can continue to run during the master node failover process.
Hadoop	High availability	HDFS NameNode and YARN ResourceManager automatically fail over to the standby node when the active master node fails.
HBase	High availability	HBase automatically fails over to the standby node when the active master node fails. If you are connecting to HBase through a REST or Thrift server, you must switch to a different master node when the active master node fails.
HCatalog	Availability not affected by master node failover	HCatalog is built upon Hive metastore, which exists outside of the cluster. HCatalog remains available during the master node failover process.
JupyterHub	High availability	JupyterHub is installed on all three master instances. It is highly recommended to configure notebook persistence to prevent notebook loss upon master node failure. For more information, see Configuring persistence for notebooks in Amazon S3.

Application	Availability during master node failover	Notes
Livy	High availability	Livy is installed on all three master nodes. When the active master node fails, you lose access to the current Livy session and need to create a new Livy session on a different master node or on the new replacement node.
Mahout	Availability not affected by master node failover	Since Mahout has no daemon, it is not affected by the master node failover process.
MXNet	Availability not affected by master node failover	Since MXNet has no daemon, it is not affected by the master node failover process.
Phoenix	High Availability	Phoenix' QueryServer runs only on one of the three master nodes. Phoenix on all three masters is configured to connect the Phoenix QueryServer. You can find the private IP of Phoenix's Query server by using `/etc/phoenix/conf/phoenix-env.sh` file
Pig	Availability not affected by master node failover	Since Pig has no daemon, it is not affected by the master node failover process.
Spark	High availability	All Spark applications run in YARN containers and can react to master node failover in the same way as high availability YARN features.
Sqoop	High availability	By default, sqoop-job and sqoop-metastore store data(job descriptions) on local disk of master that runs the command, if you want to save metastore data on external Database, please refer to apache Sqoop documentation
Tez	High availability	Since Tez containers run on YARN, Tez behaves the same way as YARN during the master node failover process.
TensorFlow	Availability not affected by master node failover	Since TensorFlow has no daemon, it is not affected by the master node failover process.
Zeppelin	High availability	Zeppelin is installed on all three master nodes. Zeppelin stores notes and interpreter configurations in HDFS by default to prevent data loss. Interpreter sessions are completely isolated across all three master instances. Session data will be lost upon master failure. It is recommended to not modify the same note concurrently on different master instances.
ZooKeeper	High availability	ZooKeeper is the foundation of the HDFS automatic failover feature. ZooKeeper provides a highly available service for maintaining coordination data, notifying clients of changes in that data, and monitoring clients for failures. For more information, see HDFS automatic failover.

To run the following applications in an Amazon EMR cluster with multiple master nodes, you must configure an external database. The external database exists outside the cluster and makes data persistent during the master node failover process. For the following applications, the service components will automatically recover during the master node failover process, but active jobs may fail and need to be retried.

Application	Availability during master node failover	Notes
Hive	High availability for service components only	An external metastore for Hive is required. This must be a MySQL external metastore, as PostgreSQL is not supported for multi-master clusters. For more information, see Configuring an external metastore for Hive.
Hue	High availability for service components only	An external database for Hue is required. For more information, see Using Hue with a remote database in Amazon RDS.
Oozie	High availability for service components only	An external database for Oozie is required. For more information, see Using Oozie with a remote database in Amazon RDS. Oozie-server and oozie-client are installed on all three master nodes. The oozie-clients are configured to connect to the correct oozie-server by default.
PrestoDB or PrestoSQL/ Trino	High availability for service components only	An external Hive metastore for PrestoDB (PrestoSQL on Amazon EMR 6.1.0-6.3.0 or Trino on Amazon EMR 6.4.0 and later) is required. You can use Presto with the AWS Glue Data Catalog or use an external MySQL database for Hive. The Presto CLI is installed on all three master nodes so you can use it to access the Presto Coordinator from any of the master nodes. The Presto Coordinator is installed on only one master node. You can find the DNS name of the master node where the Presto Coordinator is installed by calling the Amazon EMR `describe-cluster` API and reading the returned value of the `MasterPublicDnsName` field in the response.

Note
When a master node fails, your Java Database Connectivity (JDBC) or Open Database Connectivity (ODBC) terminates its connection to the master node. You can connect to any of the remaining master nodes to continue your work because the Hive metastore daemon runs on all master nodes. Or you can wait for the failed master node to be replaced.

How Amazon EMR features work in a cluster with multiple master nodes

Connecting to master nodes using SSH

You can connect to any of the three master nodes in an Amazon EMR cluster using SSH in the same way you connect to a single master node. For more information, see Connect to the master node using SSH.

If a master node fails, your SSH connection to that master node ends. To continue your work, you can connect to one of the other two master nodes. Alternatively, you can access the new master node after Amazon EMR replaces the failed one with a new one.

> **Note**
> The private IP address for the replacement master node remains the same as the previous one. The public IP address for the replacement master node may change. You can retrieve the new IP addresses in the console or by using the `describe-cluster` command in the AWS CLI. NameNode only runs on two of the master nodes. However, you can run `hdfs` CLI commands and operate jobs to access HDFS on all three master nodes.

Working with steps in an Amazon EMR Cluster with multiple master nodes

You can submit steps to an EMR cluster with multiple master nodes in the same way you work with steps in a cluster with a single master node. For more information, see Submit work to a cluster.

The following are considerations for working with steps in an EMR cluster with multiple master nodes:

- If a master node fails, the steps that are running on the master node are marked as FAILED. Any data that were written locally are lost. However, the status FAILED may not reflect the real state of the steps.
- If a running step has started a YARN application when the master node fails, the step can continue and succeed due to the automatic failover of the master node.
- It is recommended that you check the status of steps by referring to the output of the jobs. For example, MapReduce jobs use a `_SUCCESS` file to determine if the job completes successfully.
- It is recommended that you set ActionOnFailure parameter to CONTINUE, or CANCEL_AND_WAIT, instead of TERMINATE_JOB_FLOW, or TERMINATE_CLUSTER.

Automatic termination protection

Amazon EMR automatically enables termination protection for all clusters with multiple master nodes, and overrides any step execution settings that you supply when you create the cluster. You can disable termination protection after the cluster has been launched. See Configuring termination protection for running clusters (p. 175). To shut down a cluster with multiple master nodes, you must first modify the cluster attributes to disable termination protection. For instructions, see Terminate an Amazon EMR Cluster with multiple master nodes (p. 155).

For more information about termination protection, see Using termination protection (p. 172).

Unsupported features in an Amazon EMR Cluster with multiple master nodes

The following Amazon EMR features are currently not available in an EMR cluster with multiple master nodes:

- EMR Notebooks
- Instance fleets
- One-click access to persistent Spark history server
- Persistent application user interfaces

- One-click access to persistent application user interfaces is currently not available for Amazon EMR clusters with multiple master nodes or for EMR clusters integrated with AWS Lake Formation.

> **Note**
> To use Kerberos authentication in your cluster, you must configure an external KDC.
> Beginning with Amazon EMR version 5.27.0, you can configure HDFS Transparent encryption on an EMR cluster with multiple master nodes. For more information, see Transparent encryption in HDFS on Amazon EMR.

Launch an Amazon EMR Cluster with multiple master nodes

This topic provides configuration details and examples for launching an EMR cluster with multiple master nodes.

> **Note**
> Amazon EMR automatically enables termination protection for all clusters with multiple master nodes, and overrides any auto-termination settings that you supply when you create the cluster. To shut down a cluster with multiple master nodes, you must first modify the cluster attributes to disable termination protection. For instructions, see Terminate an Amazon EMR Cluster with multiple master nodes (p. 155).

Prerequisites

- You can launch an EMR cluster with multiple master nodes in both public and private VPC subnets. **EC2-Classic** is not supported. To launch an EMR cluster with multiple master nodes in a public subnet, you must enable the instances in this subnet to receive a public IP address by selecting **Auto-assign IPv4** in the console or running the following command. Replace *22XXXX01* with your subnet ID.

```
aws ec2 modify-subnet-attribute --subnet-id subnet-22XXXX01 --map-public-ip-on-launch
```

- To run Hive, Hue, or Oozie on an EMR cluster with multiple master nodes, you must create an external metastore. For more information, see Configuring an external metastore for Hive, Using Hue with a remote database in Amazon RDS, or Apache Oozie.
- To use Kerberos authentication in your cluster, you must configure an external KDC. For more information, see Configuring Kerberos on Amazon Amazon EMR.

Launch an Amazon EMR Cluster with multiple master nodes

You must specify an instance count value of three for the master node instance group when you launch an EMR cluster with multiple master nodes. The following examples demonstrate how to launch the cluster using the default AMI or a custom AMI.

> **Note**
> You must specify the subnet ID when you launch an EMR cluster with multiple master nodes using the AWS CLI. Replace *22XXXX01* with your subnet ID in the following examples.

Example – Launching an EMR cluster with multiple master nodes using a default AMI

```
aws emr create-cluster \
--name "ha-cluster" \
--release-label emr-5.36.0 \
--instance-groups InstanceGroupType=MASTER,InstanceCount=3,InstanceType=m5.xlarge
 InstanceGroupType=CORE,InstanceCount=4,InstanceType=m5.xlarge \
```

```
--ec2-attributes
 KeyName=ec2_key_pair_name,InstanceProfile=EMR_EC2_DefaultRole,SubnetId=subnet-22XXXX01 \
--service-role EMR_DefaultRole \
--applications Name=Hadoop Name=Spark
```

Example – Launching an EMR cluster with multiple master nodes using a custom AMI

```
aws emr create-cluster \
--name "custom-ami-ha-cluster" \
--release-label emr-5.36.0 \
--instance-groups InstanceGroupType=MASTER,InstanceCount=3,InstanceType=m5.xlarge
 InstanceGroupType=CORE,InstanceCount=4,InstanceType=m5.xlarge \
--ec2-attributes
 KeyName=ec2_key_pair_name,InstanceProfile=EMR_EC2_DefaultRole,SubnetId=subnet-22XXXX01 \
--service-role EMR_DefaultRole \
--applications Name=Hadoop Name=Spark \
--custom-ami-id ami-MyAmiID
```

Example – Launching an EMR cluster with multiple master nodes with an external Hive metastore

To run Hive on an EMR cluster with multiple master nodes, you must specify an external metastore for Hive, as the following example demonstrates,

1. Create a temporary hiveConfiguration.json file that contains credentials for your Hive metastore.

```
[
  {
    "Classification": "hive-site",
    "Properties": {
      "javax.jdo.option.ConnectionURL": "jdbc:mysql:\/\/hostname:3306\/hive?
createDatabaseIfNotExist=true",
      "javax.jdo.option.ConnectionDriverName": "org.mariadb.jdbc.Driver",
      "javax.jdo.option.ConnectionUserName": "username",
      "javax.jdo.option.ConnectionPassword": "password"
    }
  }
]
```

2. Launch the cluster with the Hive metastore.

```
aws emr create-cluster \
--name "ha-cluster-with-hive-metastore" \
--release-label emr-5.36.0 \
--instance-groups InstanceGroupType=MASTER,InstanceCount=3,InstanceType=m5.xlarge
 InstanceGroupType=CORE,InstanceCount=4,InstanceType=m5.xlarge \
--ec2-attributes
 KeyName=ec2_key_pair_name,InstanceProfile=EMR_EC2_DefaultRole,SubnetId=subnet-22XXXX01
 \
--service-role EMR_DefaultRole \
--applications Name=Hadoop Name= Spark Name=Hive \
--configurations ./hiveConfiguration.json
```

Terminate an Amazon EMR Cluster with multiple master nodes

To terminate an EMR cluster with multiple master nodes, you must disable termination protection before terminating the cluster, as the following example demonstrates. Replace *j-3KVTXXXXXX7UG* with your cluster ID.

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

Amazon EMR integration with EC2 placement groups

When you launch an Amazon EMR multiple master node cluster on Amazon EC2, you have the option to use placement group strategies to specify how you want the master node instances deployed to protect against hardware failure.

Placement group strategies are supported starting with Amazon EMR version 5.23.0 as an option for multiple master node clusters. Currently, only master node types are supported by the placement group strategy, and the `SPREAD` strategy is applied to those master nodes. The `SPREAD` strategy places a small group of instances across separate underlying hardware to guard against the loss of multiple master nodes in the event of a hardware failure. Note that an instance launch request could fail if there is insufficient unique hardware to fulfill the request. For more information about EC2 placement strategies and limitations, see Placement groups in the *EC2 User Guide for Linux Instances*.

There is an initial limit from Amazon EC2 of 500 placement group strategy-enabled clusters that can be launched per AWS region. Contact AWS support to request an increase in the number of allowed placement groups. You can identify EC2 placement groups Amazon EMR creates by tracking the key-value pair that Amazon EMR associates with the Amazon EMR placement group strategy. For more information about EC2 cluster instance tags, see View cluster instances in Amazon EC2 (p. 568).

Attaching the placement group managed policy to the EMR role

The placement group strategy requires a managed policy called `AmazonElasticMapReducePlacementGroupPolicy`, which allows Amazon EMR to create, delete, and describe placement groups on Amazon EC2. You must attach `AmazonElasticMapReducePlacementGroupPolicy` to the service role for Amazon EMR before you launch an Amazon EMR multiple master cluster.

You can alternatively attach the `AmazonEMRServicePolicy_v2` managed policy to the Amazon EMR service role instead of the placement group managed policy. `AmazonEMRServicePolicy_v2` allows the same access to placement groups on Amazon EC2 as the `AmazonElasticMapReducePlacementGroupPolicy`. For more information, see Service role for Amazon EMR (EMR role) (p. 398).

The `AmazonElasticMapReducePlacementGroupPolicy` managed policy is the following JSON text that is created and administered by Amazon EMR.

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

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Resource":"*",
         "Effect":"Allow",
         "Action":[
            "ec2:DeletePlacementGroup",
            "ec2:DescribePlacementGroups"
         ]
      },
      {
         "Resource":"arn:aws:ec2:*:*:placement-group/pg-*",
         "Effect":"Allow",
```

```
            "Action":[
               "ec2:CreatePlacementGroup"
            ]
         }
      ]
}
```

Launch an Amazon EMR multiple master cluster with placement group strategy

To launch an Amazon EMR multiple master cluster with a placement group strategy, attach the placement group managed policy `AmazonElasticMapReducePlacementGroupPolicy` to the Amazon EMR role. For more information, see Attaching the placement group managed policy to the EMR role (p. 156).

Every time you use this role to start an Amazon EMR multiple master cluster, Amazon EMR attempts to launch a cluster with SPREAD strategy applied to its master nodes. If you use a role that does not have the placement group managed policy `AmazonElasticMapReducePlacementGroupPolicy` attached to it, Amazon EMR attempts to launch an Amazon EMR multiple master cluster without a placement group strategy.

If you launch an Amazon EMR multiple master cluster with the `placement-group-configs` parameter using the EMR API or CLI, Amazon EMR only launches the cluster if the EMR role has the placement group managed policy `AmazonElasticMapReducePlacementGroupPolicy` attached. If the EMR role does not have the policy attached, the Amazon EMR multiple master cluster start fails.

Example – Launching an Amazon EMR multiple master cluster with placement group strategy using the EMR API.

When you use the RunJobFlow action to create an Amazon EMR multiple master cluster, set the `PlacementGroupConfigs` property to the following. Currently, the `MASTER` instance role automatically uses SPREAD as the placement group strategy.

```
{
   "Name":"ha-cluster",
   "PlacementGroupConfigs":[
      {
         "InstanceRole":"MASTER"
      }
   ],
   "ReleaseLabel":"emr-5.30.1",
   "Instances":{
      "ec2SubnetId":"subnet-22XXXX01",
      "ec2KeyName":"ec2_key_pair_name",
      "InstanceGroups":[
         {
            "InstanceCount":3,
            "InstanceRole":"MASTER",
            "InstanceType":"m5.xlarge"
         },
         {
            "InstanceCount":4,
            "InstanceRole":"CORE",
            "InstanceType":"m5.xlarge"
         }
      ]
   },
   "JobFlowRole":"EMR_EC2_DefaultRole",
   "ServiceRole":"EMR_DefaultRole"
}
```

- Replace *ha-cluster* with the name of your high-availability cluster.
- Replace *subnet-22XXXX01* with your subnet ID.
- Replace the *ec2_key_pair_name* with the name of your EC2 key pair for this cluster. EC2 key pair is optional and only required if you want to use SSH to access your cluster.

Example – Launching an Amazon EMR multiple master cluster with a placement group strategy using the EMR CLI.

```
aws emr create-cluster \
--name "ha-cluster" \
--placement-group-configs InstanceRole=MASTER \
--release-label emr-5.30.1 \
--instance-groups InstanceGroupType=MASTER,InstanceCount=3,InstanceType=m5.xlarge
 InstanceGroupType=CORE,InstanceCount=4,InstanceType=m5.xlarge \
--ec2-attributes
 KeyName=ec2_key_pair_name,InstanceProfile=EMR_EC2_DefaultRole,SubnetId=subnet-22XXXX01 \
--service-role EMR_DefaultRole \
--applications Name=Hadoop Name=Spark
```

- Replace *ha-cluster* with the name of your high-availability cluster.
- Replace *subnet-22XXXX01* with your subnet ID.
- Replace the *ec2_key_pair_name* with the name of your EC2 key pair for this cluster. EC2 key pair is optional and only required if you want to use SSH to access your cluster.

Launch an Amazon EMR multiple master cluster without a placement group strategy

For an Amazon EMR multiple master cluster to launch master nodes without the placement group strategy, you need to do one of the following:

- Remove the placement group managed policy `AmazonElasticMapReducePlacementGroupPolicy`from the EMR role, or
- Launch an Amazon EMR multiple master cluster with the `placement-group-configs` parameter using the EMR API or CLI choosing `NONE` as the placement group strategy.

Example – Launching an Amazon EMR multiple master cluster without placement group strategy using the EMR API.

When using the RunJobFlow action to create an Amazon EMR multiple master cluster, set the `PlacementGroupConfigs` property to the following.

```
{
   "Name":"ha-cluster",
   "PlacementGroupConfigs":[
      {
         "InstanceRole":"MASTER",
         "PlacementStrategy":"NONE"
      }
   ],
   "ReleaseLabel":"emr-5.30.1",
   "Instances":{
      "ec2SubnetId":"subnet-22XXXX01",
      "ec2KeyName":"ec2_key_pair_name",
      "InstanceGroups":[
         {
```

```
            "InstanceCount":3,
            "InstanceRole":"MASTER",
            "InstanceType":"m5.xlarge"
         },
         {
            "InstanceCount":4,
            "InstanceRole":"CORE",
            "InstanceType":"m5.xlarge"
         }
      ]
   },
   "JobFlowRole":"EMR_EC2_DefaultRole",
   "ServiceRole":"EMR_DefaultRole"
}
```

- Replace *ha-cluster* with the name of your high-availability cluster.
- Replace *subnet-22XXXX01* with your subnet ID.
- Replace the *ec2_key_pair_name* with the name of your EC2 key pair for this cluster. EC2 key pair is optional and only required if you want to use SSH to access your cluster.

Example – Launching an Amazon EMR multiple master cluster without a placement group strategy using the EMR CLI.

```
aws emr create-cluster \
--name "ha-cluster" \
--placement-group-configs InstanceRole=MASTER,PlacementStrategy=NONE \
--release-label emr-5.30.1 \
--instance-groups InstanceGroupType=MASTER,InstanceCount=3,InstanceType=m5.xlarge
 InstanceGroupType=CORE,InstanceCount=4,InstanceType=m5.xlarge \
--ec2-attributes
 KeyName=ec2_key_pair_name,InstanceProfile=EMR_EC2_DefaultRole,SubnetId=subnet-22XXXX01 \
--service-role EMR_DefaultRole \
--applications Name=Hadoop Name=Spark
```

- Replace *ha-cluster* with the name of your high-availability cluster.
- Replace *subnet-22XXXX01* with your subnet ID.
- Replace the *ec2_key_pair_name* with the name of your EC2 key pair for this cluster. EC2 key pair is optional and only required if you want to use SSH to access your cluster.

Checking placement group strategy configuration attached to the Amazon EMR multiple master cluster

You can use the Amazon EMR describe cluster API to see the placement group strategy configuration attached to the Amazon EMR multiple master cluster.

Example

```
aws emr describe-cluster --cluster-id "j-xxxxx"
{
   "Cluster":{
      "Id":"j-xxxxx",
      ...
      ...
      "PlacementGroups":[
         {
            "InstanceRole":"MASTER",
            "PlacementStrategy":"SPREAD"
```

```
        }
      ]
    }
}
```

Considerations and best practices

Limitations of an EMR cluster with multiple master nodes:

- If any two master nodes fail simultaneously, Amazon EMR cannot recover the cluster.
- Amazon EMR clusters with multiple master nodes are not tolerant to Availability Zone failures. In the case of an Availability Zone outage, you lose access to the Amazon EMR cluster.
- Amazon EMR does not guarantee the high availability features of open-source applications other than the ones specified in Supported applications in an Amazon EMR Cluster with multiple master nodes (p. 150).
- In Amazon EMR versions 5.23.0 up to and including 5.30.1, only two of the three master nodes run HDFS NameNode.

Considerations for configuring subnet:

- An EMR cluster with multiple master nodes can reside only in one Availability Zone or subnet. Amazon EMR cannot replace a failed master node if the subnet is fully utilized or oversubscribed in the event of a failover. To avoid this scenario, it is recommended that you dedicate an entire subnet to an EMR cluster. In addition, make sure that there are enough private IP addresses available in the subnet.

Considerations for configuring core nodes:

- To ensure the core node instance group is also highly available, it is recommended that you launch at least four core nodes. If you decide to launch a smaller cluster with three or fewer core nodes, configure HDFS with sufficient DFS replication by setting `dfs.replication parameter` to at least 2. For more information, see HDFS configuration.

Considerations for Setting Alarms on Metrics:

- Amazon EMR currently does not provide application specific metrics about HDFS or YARN. It is recommended that you set up alarms to monitor the master node instance count. You can configure the alarms using the following CloudWatch metrics: `MultiMasterInstanceGroupNodesRunning`, `MultiMasterInstanceGroupNodesRunningPercentage`, or `MultiMasterInstanceGroupNodesRequested`. You will be notified in the case of master node failure and replacement. For example,
  - If the `MultiMasterInstanceGroupNodesRunningPercentage` is lower than 1.0 and greater than 0.5, the cluster may have lost a master node. In this situation, Amazon EMR attempts to replace a master node.
  - If the `MultiMasterInstanceGroupNodesRunningPercentage` drops below 0.5, two master nodes may have failed. In this situation, the quorum is lost and the cluster cannot be recovered. Manual intervention is required to migrate data off of this cluster.

  For more information, see Setting alarms on metrics.

EMR clusters on AWS Outposts

Beginning with Amazon EMR version 5.28.0, you can create and run EMR clusters on AWS Outposts. AWS Outposts enables native AWS services, infrastructure, and operating models in on-premises facilities. In

AWS Outposts environments, you can use the same AWS APIs, tools, and infrastructure that you use in the AWS Cloud. Amazon EMR on AWS Outposts is ideal for low latency workloads that need to be run in close proximity to on-premises data and applications. For more information about AWS Outposts, see AWS Outposts User Guide.

Prerequisites

The following are the prerequisites for using Amazon EMR on AWS Outposts:

- You must have installed and configured AWS Outposts in your on-premises data center.
- You must have a reliable network connection between your Outpost environment and an AWS Region.
- You must have sufficient capacity for EMR supported instance types available in your Outpost.

Limitations

The following are the limitations of using Amazon EMR on AWS Outposts:

- On-Demand Instances are the only supported option for Amazon EC2 instances. Spot Instances are not available for Amazon EMR on AWS Outposts.
- If you need additional Amazon EBS storage volumes, only General Purpose SSD (GP2) is supported.
- S3 buckets that store objects in an AWS Region that you specify is the only supported S3 option for Amazon EMR on Outposts. S3 on Outposts is not supported for Amazon EMR on AWS Outposts.
- Only the following instance types are supported by Amazon EMR on AWS Outposts:

Instance class	Instance types
General purpose	m5.xlarge | m5.2xlarge | m5.4xlarge | m5.12xlarge | m5.24xlarge | m5d.xlarge | m5d.2xlarge | m5d.4xlarge | m5d.12xlarge | m5d.24xlarge
Compute-optimized	c5.xlarge | c5.2xlarge | c5.4xlarge | c5.9xlarge | c5.18xlarge | c5d.xlarge | c5d.2xlarge | c5d.4xlarge| c5d.9xlarge | c5d.18xlarge
Memory-optimized	r5.xlarge | r5.2xlarge | r5.4xlarge | r5.12xlarge | r5d.xlarge | r5d.2xlarge | r5d.4xlarge | r5d.12xlarge | r5d.24xlarge
Storage-optimized	i3en.xlarge | i3en.2xlarge | i3en.3xlarge | i3en.6xlarge | i3en.12xlarge | i3en.24xlarge

Network connectivity considerations

- If network connectivity between your Outpost and its AWS Region is lost, your clusters will continue to run. However, you cannot create new clusters or take new actions on existing clusters until connectivity is restored. In case of instance failures, the instance will not be automatically replaced. Additionally, actions such as adding steps to a running cluster, checking step execution status, and sending CloudWatch metrics and events will be delayed.
- We recommend that you provide reliable and highly available network connectivity between your Outpost and the AWS Region. If network connectivity between your Outpost and its AWS Region is lost for more than a few hours, clusters that have enabled terminate protection will continue to run, and clusters that have disabled terminate protection may be terminated.
- If network connectivity will be impacted due to routine maintenance, we recommend proactively enabling terminate protection. More generally, connectivity interruption means that any external

dependencies that are not local to the Outpost or customer network will not be accessible. This includes Amazon S3, DynamoDB used with EMRFS consistency view, and Amazon RDS if an in-region instance is used for an EMR cluster with multiple master nodes.

Creating an Amazon EMR cluster on AWS Outposts

Creating an Amazon EMR cluster on AWS Outposts is similar to creating an Amazon EMR cluster in the AWS Cloud. When you create an Amazon EMR cluster on AWS Outposts, you must specify an Amazon EC2 subnet associated with your Outpost.

An Amazon VPC can span all of the Availability Zones in an AWS Region. AWS Outposts are extensions of Availability Zones, and you can extend an Amazon VPC in an account to span multiple Availability Zones and associated Outpost locations. When you configure your Outpost, you associate a subnet with it to extend your Regional VPC environment to your on-premises facility. Outpost instances and related services appear as part of your Regional VPC, similar to an Availability Zone with associated subnets. For information, see AWS Outposts User Guide.

Console

To create a new Amazon EMR cluster on AWS Outposts with the AWS Management Console, specify an Amazon EC2 subnet that is associated with your Outpost.

1. Open the Amazon EMR console.
2. Choose **Create cluster**.
3. Choose **Go to advanced options**.
4. Under **Software Configuration**, for **Release**, choose 5.28.0 or later.
5. Under **Hardware Configuration**, for **EC2 Subnet**, select an EC2 subnet with an Outpost ID in this format: op-123456789.
6. Choose instance type or add Amazon EBS storage volumes for uniform instance groups or instance fleets. Limited Amazon EBS volume and instance types are supported for Amazon EMR on AWS Outposts.

AWS CLI

To create a new Amazon EMR cluster on AWS Outposts with the AWS CLI, specify an EC2 subnet that is associated with your Outpost.

The following example creates an Amazon EMR cluster on an Outpost. Replace *subnet-22XXXX01* with an EC2 subnet that is associated with your Outpost.

```
aws emr create-cluster \
--name "Outpost cluster" \
--release-label emr-5.36.0 \
--applications Name=Spark \
--ec2-attributes KeyName=myKey SubnetId=subnet-22XXXX01 \
--instance-type m5.xlarge --instance-count 3 --use-default-roles
```

EMR clusters on AWS Local Zones

Beginning with Amazon EMR version 5.28.0, you can create and run Amazon EMR clusters on an AWS Local Zones subnet as a logical extension of an AWS Region that supports Local Zones. A Local Zone enables Amazon EMR features and a subset of AWS services, like compute and storage services, to be located closer to users to provide very low latency access to applications running locally. For a list of

available Local Zones, see AWS Local Zones. For information about accessing available AWS Local Zones, see Regions, Availability Zones, and local zones.

Supported instance types

The following instance types are available for Amazon EMR clusters on Local Zones. Instance type availability may vary by Region.

Instance class	Instance types
General purpose	m5.xlarge | m5.2xlarge | m5.4xlarge | m5.12xlarge | m5.24xlarge | m5d.xlarge | m5d.2xlarge | m5d.4xlarge | m5d.12xlarge | m5d.24xlarge
Compute-optimized	c5.xlarge | c5.2xlarge | c5.4xlarge | c5.9xlarge | c5.18xlarge | c5d.xlarge | c5d.2xlarge | c5d.4xlarge| c5d.9xlarge | c5d.18xlarge
Memory-optimized	r5.xlarge | r5.2xlarge | r5.4xlarge | r5.12xlarge | r5d.xlarge | r5d.2xlarge | r5d.4xlarge | r5d.12xlarge | r5d.24xlarge
Storage-optimized	i3en.xlarge | i3en.2xlarge | i3en.3xlarge | i3en.6xlarge | i3en.12xlarge | i3en.24xlarge

Creating an Amazon EMR cluster on Local Zones

Create an Amazon EMR cluster on AWS Local Zones by launching the Amazon EMR cluster into an Amazon VPC subnet that is associated with a Local Zone. You can access the cluster using the Local Zone name, such as us-west-2-lax-1a in the US West (Oregon) Console.

Local Zones don't currently support Amazon EMR Notebooks or connections directly to Amazon EMR using interface VPC endpoint (AWS PrivateLink).

To create an EMR cluster on a Local Zone using the Amazon EMR console

To create a new Amazon EMR cluster on Local Zones with the AWS Management Console, specify an Amazon EC2 subnet that is associated with your Local Zone.

1. Open the Amazon EMR console.
2. Choose **Create cluster**.
3. Choose **Go to advanced options**.
4. Under **Software Configuration**, for **Release**, choose 5.28.0 or later.
5. Under **Hardware Configuration**, for **EC2 Subnet**, select an EC2 subnet with a Local Zone ID in this format: subnet 123abc | us-west-2-lax-1a.
6. Add Amazon EBS storage volumes for uniform instance groups or instance fleets and choose an instance type.

To create an Amazon EMR cluster on a Local Zone using the AWS CLI

Use the create-cluster command, along with the SubnetId for the Local Zone as shown in the following example. Replace subnet-22XXXX1234567 with the Local Zone SubnetId and replace other options as necessary. For more information, see https://docs.aws.amazon.com/cli/latest/reference/emr/create-cluster.html.

```
aws emr create-cluster \
```

```
--name "Local Zones cluster" \
--release-label emr-5.29.0 \
--applications Name=Spark \
--ec2-attributes KeyName=myKey,SubnetId=subnet-22XXXX1234567 \
--instance-type m5.xlarge --instance-count 3 --use-default-roles
```

Configure Docker

Amazon EMR 6.x supports Hadoop 3, which allows the YARN NodeManager to launch containers either directly on the Amazon EMR cluster or inside a Docker container. Docker containers provide custom execution environments in which application code runs. The custom execution environment is isolated from the execution environment of the YARN NodeManager and other applications.

Docker containers can include special libraries used by the application and they can provide different versions of native tools and libraries, such as R and Python. You can use familiar Docker tooling to define libraries and runtime dependencies for your applications.

Amazon EMR 6.x clusters are configured by default to allow YARN applications, such as Spark, to run using Docker containers. To customize your container configuration, edit the Docker support options defined in the `yarn-site.xml` and `container-executor.cfg` files available in the `/etc/hadoop/conf` directory. For details about each configuration option and how it is used, see Launching applications using Docker containers.

You can choose to use Docker when you submit a job. Use the following variables to specify the Docker runtime and Docker image.

- `YARN_CONTAINER_RUNTIME_TYPE=docker`
- `YARN_CONTAINER_RUNTIME_DOCKER_IMAGE={`*`DOCKER_IMAGE_NAME`*`}`

When you use Docker containers to run your YARN applications, YARN downloads the Docker image that you specify when you submit your job. For YARN to resolve this Docker image, it must be configured with a Docker registry. The configuration options for a Docker registry depend on whether you deploy the cluster using a public or private subnet.

Docker registries

A Docker registry is a storage and distribution system for Docker images. For Amazon EMR we recommend that you use Amazon ECR, which is a fully managed Docker container registry that allows you to create your own custom images and host them in a highly available and scalable architecture.

Deployment considerations

Docker registries require network access from each host in the cluster. This is because each host downloads images from the Docker registry when your YARN application is running on the cluster. These network connectivity requirements may limit your choice of Docker registry, depending on whether you deploy your Amazon EMR cluster into a public or private subnet.

Public subnet

When EMR clusters are deployed in a public subnet, the nodes running YARN NodeManager can directly access any registry available over the internet.

Private subnet

When EMR clusters are deployed in a private subnet, the nodes running YARN NodeManager don't have direct access to the internet. Docker images can be hosted in Amazon ECR and accessed through AWS PrivateLink.

For more information about how to use AWS PrivateLink to allow access to Amazon ECR in a private subnet scenario, see Setting up AWS PrivateLink for Amazon ECS, and Amazon ECR.

Configuring Docker registries

To use Docker registries with Amazon EMR, you must configure Docker to trust the specific registry that you want to use to resolve Docker images. The default trust registries are local (private) and centos. To use other public repositories or Amazon ECR, you can override `docker.trusted.registries` settings in `/etc/hadoop/conf/container-executor.cfg` using the EMR Classification API with the `container-executor` classification key.

The following example shows how to configure the cluster to trust both a public repository, named `your-public-repo`, and an ECR registry endpoint, `123456789123.dkr.ecr.us-east-1.amazonaws.com`. If you use ECR, replace this endpoint with your specific ECR endpoint.

```
[
  {
    "Classification": "container-executor",
    "Configurations": [
        {
            "Classification": "docker",
            "Properties": {
                "docker.trusted.registries": "local,centos,your-public-
repo,123456789123.dkr.ecr.us-east-1.amazonaws.com",
                "docker.privileged-containers.registries": "local,centos,your-public-
repo,123456789123.dkr.ecr.us-east-1.amazonaws.com"
            }
        }
    ]
  }
]
```

To launch an Amazon EMR 6.0.0 cluster with this configuration using the AWS Command Line Interface (AWS CLI), create a file named `container-executor.json` with the contents of the preceding ontainer-executor JSON configuration. Then, use the following commands to launch the cluster.

```
export KEYPAIR=<Name of your Amazon EC2 key-pair>
export SUBNET_ID=<ID of the subnet to which to deploy the cluster>
export INSTANCE_TYPE=<Name of the instance type to use>
export REGION=<Region to which to deploy the cluster>

aws emr create-cluster \
    --name "EMR-6.0.0" \
    --region $REGION \
    --release-label emr-6.0.0 \
    --applications Name=Hadoop Name=Spark \
    --service-role EMR_DefaultRole \
    --ec2-attributes KeyName=$KEYPAIR,InstanceProfile=EMR_EC2_DefaultRole,SubnetId=
$SUBNET_ID \
    --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=$INSTANCE_TYPE
 InstanceGroupType=CORE,InstanceCount=2,InstanceType=$INSTANCE_TYPE \
    --configuration file://container-executor.json
```

Configuring YARN to access Amazon ECR on EMR 6.0.0 and earlier

If you're new to Amazon ECR, follow the instructions in Getting started with Amazon ECR and verify that you have access to Amazon ECR from each instance in your Amazon EMR cluster.

On EMR 6.0.0 and earlier, to access Amazon ECR using the Docker command, you must first generate credentials. To verify that YARN can access images from Amazon ECR, use the container environment variable `YARN_CONTAINER_RUNTIME_DOCKER_CLIENT_CONFIG` to pass a reference to the credentials that you generated.

Run the following command on one of the core nodes to get the login line for your ECR account.

```
aws ecr get-login --region us-east-1 --no-include-email
```

The `get-login` command generates the correct Docker CLI command to run to create credentials. Copy and run the output from `get-login`.

```
sudo docker login -u AWS -p <password> https://<account-id>.dkr.ecr.us-east-1.amazonaws.com
```

This command generates a `config.json` file in the `/root/.docker` folder. Copy this file to HDFS so that jobs submitted to the cluster can use it to authenticate to Amazon ECR.

Run the commands below to copy the `config.json` file to your home directory.

```
mkdir -p ~/.docker
sudo cp /root/.docker/config.json ~/.docker/config.json
sudo chmod 644 ~/.docker/config.json
```

Run the commands below to put the config.json in HDFS so it may be used by jobs running on the cluster.

```
hadoop fs -put ~/.docker/config.json /user/hadoop/
```

YARN can access ECR as a Docker image registry and pull containers during job execution.

After configuring Docker registries and YARN, you can run YARN applications using Docker containers. For more information, see Run Spark applications with Docker using Amazon EMR 6.0.0.

In EMR 6.1.0 and later, you don't have to manually set up authentication to Amazon ECR. If an Amazon ECR registry is detected in the `container-executor` classification key, the Amazon ECR auto authentication feature activates, and YARN handles the authentication process when you submit a Spark job with an ECR image. You can confirm whether automatic authentication is enabled by checking `yarn.nodemanager.runtime.linux.docker.ecr-auto-authentication.enabled` in yarn-site. Automatic authentication is enabled and the YARN authentication setting is set to `true` if the `docker.trusted.registries` contains an ECR registry URL.

Prerequisites for using automatic authentication to Amazon ECR

- EMR version 6.1.0 or later
- ECR registry included in configuration is in the same Region with the cluster
- IAM role with permissions to get authorization token and pull any image

Refer to Setting up with Amazon ECR for more information.

How to enable automatic authentication

Follow Configuring Docker registries (p. 165) to set an Amazon ECR registry as a trusted registry, and make sure the Amazon ECR repository and the cluster are in same Region.

To enable this feature even when the ECR registry is not set in the trusted registry, use the configuration classification to set `yarn.nodemanager.runtime.linux.docker.ecr-auto-authentication.enabled` to `true`.

How to disable automatic authentication

By default, automatic authentication is disabled if no Amazon ECR registry is detected in the trusted registry.

To disable automatic authentication, even when the Amazon ECR registry is set in the trusted registry, use the configuration classification to set `yarn.nodemanager.runtime.linux.docker.ecr-auto-authentication.enabled` to `false`.

How to check if automatic authentication is enabled on a cluster

On the master node, use a text editor such as `vi` to view the contents of the file: `vi /etc/hadoop/conf.empty/yarn-site.xml`. Check the value of `yarn.nodemanager.runtime.linux.docker.ecr-auto-authentication.enabled`.

Control cluster termination

This section describes your options for shutting down Amazon EMR clusters. It covers auto-termination and termination protection, and how they interact with other Amazon EMR features.

You can shut down an Amazon EMR cluster in the following ways:

- **Termination after last step execution** - Create a transient cluster that shuts down after all steps complete.
- **Auto-termination (after idle)** - Create a cluster with an auto-termination policy that shuts down after a specified idle time. For more information, see Using an auto-termination policy (p. 169).
- **Manual termination** - Create a long-running cluster that continues to run until you terminate it deliberately. For information about how to terminate a cluster manually, see Terminate a cluster (p. 607).

You can also set termination protection on a cluster to avoid shutting down EC2 instances by accident or error.

When Amazon EMR shuts down your cluster, all Amazon EC2 instances in the cluster shut down. 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.

Topics

- Configuring a cluster to continue or terminate after step execution (p. 167)
- Using an auto-termination policy (p. 169)
- Using termination protection (p. 172)

Configuring a cluster to continue or terminate after step execution

This topic explains the differences between using a long-running cluster and creating a transient cluster that shuts down after the last step runs. It also covers how to configure step execution for a cluster.

Create a long-running cluster

By default, clusters that you create using the console or the AWS CLI are long-running. Long-running clusters continue to run, accept work, and accrue charges until you take action to shut them down.

A long-running cluster is effective in the following situations:

- When you need to interactively or automatically query data.
- When you need to interact with big data applications hosted on the cluster on an ongoing basis.
- When 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 also set termination protection on a long-running cluster to avoid shutting down EC2 instances by accident or error. For more information, see Using termination protection (p. 172).

> **Note**
> Amazon EMR automatically enables termination protection for all clusters with multiple master nodes, and overrides any step execution settings that you supply when you create the cluster. You can disable termination protection after the cluster has been launched. See Configuring termination protection for running clusters (p. 175). To shut down a cluster with multiple master nodes, you must first modify the cluster attributes to disable termination protection. For instructions, see Terminate an Amazon EMR Cluster with multiple master nodes (p. 155).

Configure a cluster to terminate after step execution

When you configure termination after step execution, the cluster starts, runs bootstrap actions, and then runs the steps that you specify. As soon as the last step completes, Amazon EMR terminates the cluster's Amazon EC2 instances. Clusters that you launch using the Amazon EMR API have step execution enabled by default.

Termination after step execution is effective for clusters that perform a periodic processing task, such as a daily data processing run. Step execution also helps you ensure that you are billed only for the time required to process your data. For more information about steps, see Submit work to a cluster (p. 642).

To disable step execution using the Amazon EMRAPI

- When using the RunJobFlow action to create a cluster, set the KeepJobFlowAliveWhenNoSteps property to `true`.

To enable step execution using quick options in the AWS Management Console

1. Open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Choose **Create cluster**.
3. Choose **Step execution**.
4. Choose other settings as appropriate for your application, and then choose **Create cluster**.

To enable step execution using Advanced Options in the AWS Management Console

1. Open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Choose **Create cluster**.
3. Choose **Go to advanced options**.
4. Under **Add steps (optional)** select **Auto-terminate cluster after the last step is completed**.

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

To enable step execution 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-5.36.0 \
--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 m5.xlarge --instance-count 3 --auto-terminate
```

Using an auto-termination policy

An auto-termination policy lets you orchestrate cluster cleanup without the need to monitor and manually terminate unused clusters. When you add an auto-termination policy to a cluster, you specify the amount of idle time after which the cluster should automatically shut down.

Depending on release version, Amazon EMR uses different criteria to mark a cluster as idle. The following table outlines how Amazon EMR determines cluster idleness.

When you use ...	A cluster is considered idle when ...
Amazon EMR versions 6.4.0 and later	• There are no active YARN applications • HDFS utilization is below 10% • There are no active EMR notebook or EMR Studio connections • There are no on-cluster application user interfaces in use
Amazon EMR versions 5.30.0 - 5.33.0 and 6.1.0 - 6.3.0	• There are no active YARN applications • HDFS utilization is below 10% • The cluster has no active Spark jobs **Note** Amazon EMR marks a cluster as idle and may automatically terminate the cluster even if you have an active Python3 kernel. This is because executing a Python3 kernel does not submit a Spark job on the cluster. To use auto-termination with a Python3 kernel, we recommend that you use Amazon EMR version 6.4.0 or later.

> **Note**
> Amazon EMR versions 6.4.0 and later support an on-cluster file for detecting activity on the master node: `/emr/metricscollector/isbusy`. When you use a cluster to run shell scripts or non-YARN applications, you can periodically touch or update `isbusy` to tell Amazon EMR that the cluster is not idle.

You can attach an auto-termination policy when you create a cluster, or add a policy to an existing cluster. To change or disable auto-termination, you can update or remove the policy.

Considerations

Consider the following features and limitations before using an auto-termination policy:

- Auto-termination is supported with Amazon EMR versions 5.30.0 and 6.1.0 and later.
- Auto-termination is available in the following AWS Regions: US East (N. Virginia, Ohio), US West (N. California, Oregon), Asia Pacific (Mumbai, Seoul, Singapore, Sydney, Tokyo, Hong Kong), Canada (Central), China (Beijing, Ningxia), Europe (Ireland, Frankfurt, London, Paris, Stockholm, Milan), South America (São Paulo), Middle East (Bahrain), and Africa (Cape Town).
- Idle timeout defaults to 60 minutes (one hour) when you don't specify an amount. You can specify a minimum idle timeout of one minute, and a maximum idle timeout of 7 days.
- With Amazon EMR versions 6.4.0 and later, auto-termination is enabled by default when you create a new cluster using the Amazon EMR console.
- Amazon EMR publishes high-resolution Amazon CloudWatch metrics when you enable auto-termination for a cluster. You can use these metrics to track cluster activity and idleness. For more information, see Cluster capacity metrics (p. 587).
- Auto-termination is not supported when you use non-YARN based applications such as Presto, Trino, or HBase.
- To use auto-termination, the metrics-collector process must be able to connect to the public API endpoint for auto-termination in API Gateway. If you use a private DNS name with Amazon Virtual Private Cloud, auto-termination won't function properly. To ensure that auto-termination works, we recommend that you take one of the following actions:
  - Remove the API Gateway interface VPC endpoint from your Amazon VPC.
  - Follow the instructions in Why do I get an HTTP 403 Forbidden error when connecting to my API Gateway APIs from a VPC? to disable the private DNS name setting.
  - Launch your cluster in a private subnet instead. For more information, see the topic on Private subnets (p. 308).
- (EMR 5.30.0 and later) If you remove the default **Allow All** outbound rule to 0.0.0.0/ for the master security group, you must add a rule that allows outbound TCP connectivity to your security group for service access on port 9443. Your security group for service access must also allow inbound TCP traffic on port 9443 from the master security group. For more information about configuring security groups, see Amazon EMR-managed security group for the master instance (private subnets).

Permissions to use auto-termination

Before you can apply and manage auto-termination policies for Amazon EMR, you need to attach the permissions that are listed in the following example IAM permissions policy to your cluster role.

```
{
  "Version": "2012-10-17",
  "Statement": {
      "Sid": "AllowAutoTerminationPolicyActions",
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:PutAutoTerminationPolicy",
```

```
        "elasticmapreduce:GetAutoTerminationPolicy",
        "elasticmapreduce:RemoveAutoTerminationPolicy"
      ],
      "Resource": "<your-resources>"
    }
}
```

Attach, update, or remove an auto-termination policy

This section includes instructions to help you attach, update, or remove an auto-termination policy from an Amazon EMR cluster. Before you work with auto-termination policies, make sure you have the necessary IAM permissions. See Permissions to use auto-termination (p. 170).

Console

To attach an auto-termination policy when you create a cluster

1. Open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Choose **Create cluster**.
3. Under **Hardware configuration**, select **Auto-termination**.
4. Specify the number of idle hours and minutes after which the cluster should auto-terminate. The default idle time is one hour.
5. Choose other settings as appropriate for your application, and then choose **Create cluster**.

To attach an auto-termination policy when you create an EMR notebook

1. Open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Choose **Notebooks**, then choose **Create notebook**.
3. Choose **Create a cluster**.
4. Select **Enable auto-termination**, then specify the number of idle hours and minutes after which the cluster should auto-terminate. The default idle time is one hour.
5. Choose other settings as appropriate for your EMR notebook, and then choose **Create notebook**.

To attach, update, or remove an auto-termination policy on a running cluster

1. Open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Select **Clusters** and choose the cluster you want to update.
3. Choose the **Hardware** tab on the cluster detail page.
4. Select or clear **Enable auto-termination** to enable or disable the feature. If you enable auto-termination, specify the number of idle hours and minutes after which the cluster should auto-terminate.

AWS CLI

Before you start

Before you work with auto-termination policies, we recommend that you update to the latest version of the AWS CLI. For instructions, see Installing, updating, and uninstalling the AWS CLI.

To attach or update an auto-termination policy using the AWS CLI

- You can use the `aws emr put-auto-termination-policy` command to attach or update an auto-termination policy on a cluster.

The following example specifies 3600 seconds for *IdleTimeout*. If you don't specify *IdleTimeout*, the value defaults to one hour.

```
aws emr put-auto-termination-policy \
--cluster-id <your-cluster-id> \
--auto-termination-policy IdleTimeout=3600
```

> **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 (^).

You can also specify a value for `--auto-termination-policy` when you use the `aws emr create-cluster` command. For more information on using Amazon EMR commands in the AWS CLI, see the AWS CLI Command Reference.

To remove an auto-termination policy using the AWS CLI

- Use the `aws emr remove-auto-termination-policy` command to remove an auto-termination policy from a cluster. For more information on using Amazon EMR commands in the AWS CLI, see the AWS CLI Command Reference.

```
aws emr remove-auto-termination-policy --cluster-id <your-cluster-id>
```

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. 634).
- Removing instances from a core or task instance group using a scale-in policy with automatic scaling. For more information, see Using automatic scaling with a custom policy for instance groups (p. 625).

- Removing instances from an instance fleet by reducing target capacity. For more information, see Instance fleet options (p. 317).

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 deny lists 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. 666).

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)	Amazon EC2 core instances remain in a deny-listed state and continue to count toward cluster capacity. You can connect to an Amazon EC2 core instance for configuration and data recovery, and resize your cluster to add capacity. For more information, see Resource errors (p. 666). Unhealthy task nodes are exempt from termination protection and will be terminated.
Disabled	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. **Important** HDFS data may be lost if a core instance terminates because of an unhealthy state. If the node stored blocks that were

Termination protection	Result
	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.

Termination protection and step execution

When you enable step execution and *also* enable termination protection, Amazon EMR ignores the termination protection.

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 Submit work to a cluster (p. 642).

If a step fails that is configured with `ActionOnFailure` set to `CANCEL_AND_WAIT`, if step execution 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	Step execution	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

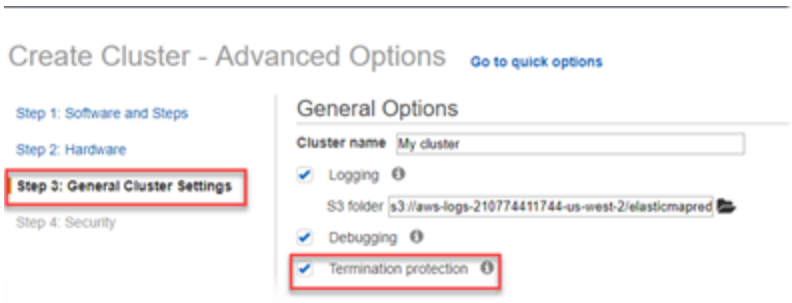
You can enable or disable termination protection when you launch a cluster using the console, the AWS CLI, or the API.

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/emr.
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-5.36.0
 \
--applications Name=Hadoop Name=Hive Name=Pig \
--use-default-roles --ec2-attributes KeyName=myKey --instance-type m5.xlarge \
--instance-count 3 --termination-protected
```

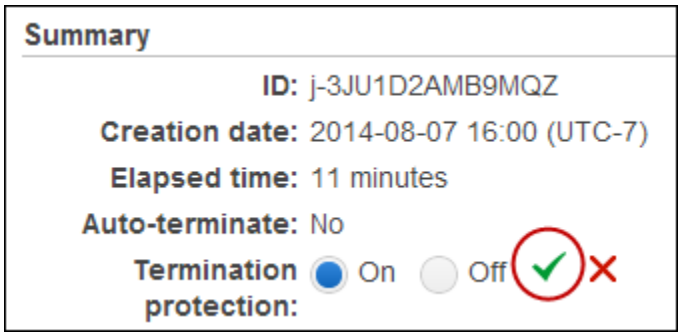
  For more information about using Amazon EMR commands in the AWS CLI, see https://docs.aws.amazon.com/cli/latest/reference/emr.

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/emr.
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. For more information about the default Amazon Linux AMI, see Using the default Amazon Linux AMI for Amazon EMR (p. 177). 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. You can specify a custom AMI for each instance type in the instance group or instance fleet configuration of an Amazon EMR cluster. Multiple custom AMI support gives you the flexibility to use more than one architecture type in a cluster. See Using a custom AMI (p. 181).

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 Amazon 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 24 hours times 10 GB). Whether you use the default Amazon Linux AMI or a custom Amazon Linux AMI, you can specify the size of the Amazon 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 storage (p. 303).

Topics

- Using the default Amazon Linux AMI for Amazon EMR (p. 177)
- Using a custom AMI (p. 181)
- Changing the Amazon Linux release for a launching cluster (p. 188)
- Specifying the Amazon EBS root device volume size (p. 189)

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 behavior for updating Amazon Linux 2 (AL2) in an Amazon EMR default AMI is different starting with Amazon EMR 5.36.0 and later, and Amazon EMR 6.6.0 and later releases.

Amazon EMR 5.36.0 and later, and 6.6.0 and later — automatic Amazon Linux updates

When you launch a cluster using Amazon EMR 5.36.0 and later, or Amazon EMR 6.6.0 and later, it automatically uses the latest Amazon Linux 2 release that has been validated for the default Amazon EMR AMI. This is the default behavior for Amazon EMR version 5.36.0 and later, and Amazon EMR versions 6.6.0 and later. For specific release versions, see Amazon EMR 5.x release versions or Amazon EMR 6.x release versions in the *Amazon EMR Release Guide*. You have the option of launching the cluster with the original Amazon Linux version that Amazon EMR versions 5.36.0 and later, and 6.6.0 and later, first shipped with. For information on specifying the Amazon Linux release for your cluster, see Changing the Amazon Linux release for a launching cluster (p. 188).

The following table lists Amazon Linux information for Amazon EMR version 6.6.0.

OsReleas (Amazon Linux Version)	Amazon Linux Kernel Version	Available Date	Supported Regions
2.0.20220406	4.14.275	5/2/2022	us-east-1, us-east-2, us-west-1, us-west-2, eu-north-1, eu-west-1, eu-west-2, eu-west-3, eu-central-1, eu-south-1, ap-east-1, ap-south-1, ap-southeast-3, ap-northeast-1, ap-northeast-2, ap-northeast-3, ap-southeast-1, ap-southeast-2, af-south-1, sa-east-1, me-south-1, ca-central-1
2.0.20220426	4.14.281	6/10/2022	us-east-1, us-east-2, us-west-1, us-west-2, eu-north-1, eu-west-1, eu-west-2, eu-west-3, eu-central-1, eu-south-1,

OsRelease (Amazon Linux Version)	Amazon Linux Kernel Version	Available Date	Supported Regions
			ap-east-1, ap-south-1, ap-southeast-3, ap-northeast-1, ap-northeast-2, ap-northeast-3, ap-southeast-1, ap-southeast-2, af-south-1, sa-east-1, me-south-1, ca-central-1
2.0.20220719.0	4.14.287	8/10/2022	us-east-1, us-east-2, us-west-1, us-west-2, eu-north-1, eu-west-1, eu-west-2, eu-west-3, eu-central-1, eu-south-1, ap-east-1, ap-south-1, ap-southeast-3, ap-northeast-1, ap-northeast-2, ap-northeast-3, ap-southeast-1, ap-southeast-2, af-south-1, sa-east-1, me-south-1, ca-central-1
2.0.20220830.0	4.14.287	8/30/2022	us-west-1
2.0.20220912.1	4.14.291	10/7/2022	us-east-1, us-east-2, us-west-1, us-west-2, eu-north-1, eu-west-1, eu-west-2, eu-west-3, eu-central-1, eu-south-1, ap-east-1, ap-south-1, ap-southeast-3, ap-northeast-1, ap-northeast-2, ap-northeast-3, ap-southeast-1, ap-southeast-2, af-south-1, sa-east-1, me-south-1, ca-central-1

The following table lists Amazon Linux information for Amazon EMR version 5.36.0.

OsReleas (Amazon Linux Version)	Amazon Linux Kernel Version	Available Date	Supported Regions
2.0.20220426.0	4.14.281	6/14/2022	us-east-1, us-east-2, us-west-1, us-west-2, eu-north-1, eu-west-1, eu-west-2, eu-west-3, eu-central-1, eu-south-1, ap-east-1, ap-south-1, ap-southeast-3, ap-northeast-1, ap-northeast-2, ap-northeast-3, ap-southeast-1, ap-southeast-2, af-south-1, sa-east-1, me-south-1, ca-central-1
2.0.20220719.0	4.14.287	8/10/2022	us-west-1, eu-west-3, eu-north-1, eu-central-1, ap-south-1, me-south-1
2.0.20220912.1	4.14.291	10/7/2022	us-east-1, us-east-2, us-west-1, us-west-2, eu-north-1, eu-west-1, eu-west-2, eu-west-3, eu-central-1, eu-south-1, ap-east-1, ap-south-1, ap-southeast-3, ap-northeast-1, ap-northeast-2, ap-northeast-3, ap-southeast-1, ap-southeast-2, af-south-1, sa-east-1, me-south-1, ca-central-1

Amazon EMR 5.35.0 and earlier, and 6.5.0 and earlier — Amazon Linux AMI “locked” to Amazon EMR release version

For Amazon EMR 5.35.0 and earlier, and 6.5.0 and earlier, 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 5.35.0 and earlier, and 6.5.0 and earlier Amazon EMR release version is "locked" to its respective assigned Amazon Linux AMI version to maintain compatibility. This means that earlier Amazon Linux AMI versions are used for earlier Amazon EMR release versions, even when newer Amazon Linux AMIs become available. For this reason, we recommend that you use the latest Amazon EMR release

version, 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.

Managing software updates

Here are default software update behaviors to be aware of:

- **Default boot behavior excludes kernel updates**. 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. However, if you are using an older version of Amazon Linux AMI, the latest security update may not be installed automatically. This is because the repositories referenced by EMR are fixed for each version of Amazon Linux AMI. Also note that your networking configuration must allow for HTTP and HTTPS egress to Amazon Linux repositories in Amazon S3, otherwise security updates will not succeed. For more information, see Package repository in the *Amazon EC2 User Guide for Linux Instances*. By default, other software packages and kernel updates that require a reboot, including NVIDIA and CUDA, are excluded from the automatic download at first boot.

  > **Important**
  > Amazon EMR clusters that are running Amazon Linux or Amazon Linux 2 AMIs (Amazon Linux Machine Images) use default Amazon Linux behavior, and do not automatically download and install important and critical kernel updates that require a reboot. This is the same behavior as other Amazon EC2 instances running the default Amazon Linux AMI. If new Amazon Linux software updates that require a reboot (such as, kernel, NVIDIA, and CUDA updates) become available after an Amazon EMR version is released, Amazon EMR cluster instances running the default AMI do not automatically download and install those updates. To get kernel updates, you can customize your Amazon EMR AMI to use the latest Amazon Linux AMI.

- **The cluster launches with or without updates**. Be aware that if software updates can't be installed because package repositories are unreachable at first cluster boot, the cluster instance still completes its launch. For instance, repositories may be unreachable because S3 is temporarily unavailable, or you might have VPC or firewall rules configured to block access.
- **Don't run `sudo yum update`**. 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 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. 191).

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:

- 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. 181).
- Implement more sophisticated cluster and node configurations than bootstrap actions allow.
- Encrypt the EBS root device volumes (boot volumes) of EC2 instances in your cluster if you are using an Amazon EMR version earlier than 5.24.0. As with the default AMI, the minimum root volume size for a custom AMI is 10 GiB. For more information, see Creating a custom AMI with an encrypted Amazon EBS root device volume (p. 186).

  Note
  Beginning with Amazon EMR version 5.24.0, you can use a security configuration option to encrypt EBS root device and storage volumes when you specify AWS KMS as your key provider. For more information, see Local disk encryption (p. 376).

A custom AMI must exist in the same AWS Region where you create the cluster. It should also match the EC2 instance architecture. For example, an m5.xlarge instance has an x86_64 architecture. Therefore, to provision an m5.xlarge using a custom AMI, your custom AMI should also have x86_64 architecture. Similarly, to provision an m6g.xlarge instance, which has arm64 architecture, your custom AMI should have arm64 architecture. For more information about identifying a Linux AMI for your instance type, see Find a Linux AMI in the *Amazon EC2 User Guide for Linux Instances*.

Important
Amazon EMR clusters that are running Amazon Linux or Amazon Linux 2 AMIs (Amazon Linux Machine Images) use default Amazon Linux behavior, and do not automatically download and install important and critical kernel updates that require a reboot. This is the same behavior as other Amazon EC2 instances running the default Amazon Linux AMI. If new Amazon Linux software updates that require a reboot (such as, kernel, NVIDIA, and CUDA updates) become available after an Amazon EMR version is released, Amazon EMR cluster instances running the default AMI do not automatically download and install those updates. To get kernel updates, you can customize your Amazon EMR AMI to use the latest Amazon Linux AMI.

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. 186).

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 m5.xlarge \
   --key-name MyKeyName --region us-west-2
   ```

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

2. Run the following command:

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

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

To connect to the instance and install software

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

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

To create a snapshot from your customized image

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

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

  The `imageID` output value is used when you launch the cluster or create an encrypted snapshot. For more information, see Use a single custom AMI in an EMR cluster (p. 183) and Creating a custom AMI with an encrypted Amazon EBS root device volume (p. 186).

How to use a custom AMI in an Amazon EMR cluster

You can use a custom AMI to provision an Amazon EMR cluster in two ways:

- Use a single custom AMI for all the EC2 instances in the cluster.
- Use different custom AMIs for the different EC2 instance types used in the cluster.

You can use only one of the two options when provisioning an EMR cluster, and you cannot change it once the cluster has started.

Considerations for using single versus multiple custom AMIs in an Amazon EMR cluster

Consideration	Single custom AMI	Multiple custom AMIs
Use both x86 and Graviton2 processors with custom AMIs in the same cluster	✕ Not supported	✓ Supported
AMI customization varies across instance types	✕ Not supported	✓ Supported
Change custom AMIs when adding new task instance groups/fleets to a running cluster. Note: you cannot change the custom AMI of existing instance groups/fleets.	✕ Not supported	✓ Supported
Use AWS Console to start a cluster	✓ Supported	✕ Not supported
Use AWS CloudFormation to start a cluster	✓ Supported	✓ Supported

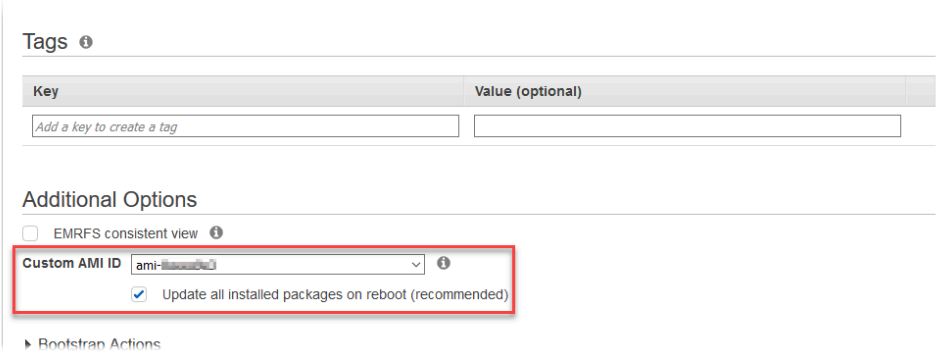
Use a single custom AMI in an EMR cluster

To specify a custom AMI ID when you create a cluster, use one of the following:

- AWS Management Console
- AWS CLI
- Amazon EMR SDK
- Amazon EMR API RunJobFlow
- AWS CloudFormation (see the `CustomAmiID` property in Cluster InstanceGroupConfig, Cluster InstanceTypeConfig, Resource InstanceGroupConfig, or Resource InstanceFleetConfig-InstanceTypeConfig)

To use a single custom AMI in the console

1. Open the Amazon EMR console at https://console.aws.amazon.com/emr.
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 make sure that the **Update all installed packages on reboot** option is selected. For more information about changing the update option, see Managing AMI package repository updates (p. 186).



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

To specify a single 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 single custom AMI with a 20 GiB boot volume. For more information, see Specifying the Amazon EBS root device volume size (p. 189).

  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 m5.xlarge
```

Use multiple custom AMIs in an Amazon EMR cluster

To create a cluster using multiple custom AMIs, use one of the following:

- AWS CLI version 1.20.21 or later
- AWS SDK
- Amazon EMR RunJobFlow in the *Amazon EMR API Reference*
- AWS CloudFormation (see the `CustomAmiID` property in Cluster InstanceGroupConfig, Cluster InstanceTypeConfig, Resource InstanceGroupConfig, or Resource InstanceFleetConfig-InstanceTypeConfig)

The AWS Management Console currently does not support creating a cluster using multiple custom AMIs.

Example - Use the AWS CLI to create an instance group cluster using multiple custom AMIs

Using the AWS CLI version 1.20.21 or later, you can assign a single custom AMI to the entire cluster, or you can assign multiple custom AMIs to every instance node in your cluster.

The following example shows a uniform instance group cluster created with two instance types (m5.xlarge) used across node types (master, core, task). Each node has multiple custom AMIs. The example illustrates several features of the multiple custom AMI configuration:

- There is no custom AMI assigned at the cluster level. This is to avoid conflicts between the multiple custom AMIs and a single custom AMI, which would cause the cluster launch to fail.
- The cluster can have multiple custom AMIs across master, core, and individual task nodes. This allows individual AMI customizations, such as pre-installed applications, sophisticated cluster configurations, and encrypted Amazon EBS root device volumes.
- The instance group core node can have only one instance type and corresponding custom AMI. Similarly, the master node can have only one instance type and corresponding custom AMI.
- The cluster can have multiple task nodes.

```
aws emr create-cluster --instance-groups
InstanceGroupType=MASTER,InstanceType=m5.xlarge,InstanceCount=1,CustomAmiId=ami-123456
InstanceGroupType=CORE,InstanceType=m5.xlarge,InstanceCount=1,CustomAmiId=ami-234567
InstanceGroupType=TASK,InstanceType=m6g.xlarge,InstanceCount=1,CustomAmiId=ami-345678
InstanceGroupType=TASK,InstanceType=m5.xlarge,InstanceCount=1,CustomAmiId=ami-456789
```

Example - Use the AWS CLI version 1.20.21 or later to add a task node to a running instance group cluster with multiple instance types and multiple custom AMIs

Using the AWS CLI version 1.20.21 or later, you can add multiple custom AMIs to an instance group that you add to a running cluster. The `CustomAmiId` argument can be used with the `add-instance-groups` command as shown in the following example. Notice that the same multiple custom AMI ID (ami-123456) is used in more than one node.

```
aws emr create-cluster --instance-groups
InstanceGroupType=MASTER,InstanceType=m5.xlarge,InstanceCount=1,CustomAmiId=ami-123456
InstanceGroupType=CORE,InstanceType=m5.xlarge,InstanceCount=1,CustomAmiId=ami-123456
InstanceGroupType=TASK,InstanceType=m5.xlarge,InstanceCount=1,CustomAmiId=ami-234567

{
    "ClusterId": "j-123456",
    ...
}

aws emr add-instance-groups --cluster-id j-123456 --instance-groups
 InstanceGroupType=Task,InstanceType=m6g.xlarge,InstanceCount=1,CustomAmiId=ami-345678
```

Example - Use the AWS CLI version 1.20.21 or later to create an instance fleet cluster, multiple custom AMIs, multiple instance types, On-Demand master, On-Demand core, multiple core and task nodes

```
aws emr create-cluster --instance-fleets
InstanceFleetType=MASTER,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=m5.xlarge,
 CustomAmiId=ami-123456}']
InstanceFleetType=CORE,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=m5.xlarge,CustomAmi
{InstanceType=m6g.xlarge, CustomAmiId=ami-345678}']
InstanceFleetType=TASK,TargetSpotCapacity=1,InstanceTypeConfigs=['{InstanceType=m5.xlarge,CustomAmiId=a
{InstanceType=m6g.xlarge, CustomAmiId=ami-567890}']
```

Example - Use the AWS CLI version 1.20.21 or later to add task nodes to a running cluster with multiple instance types and multiple custom AMIs

```
aws emr create-cluster --instance-fleets
InstanceFleetType=MASTER,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=m5.xlarge,
 CustomAmiId=ami-123456}']
InstanceFleetType=CORE,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=m5.xlarge,CustomAmi
{InstanceType=m6g.xlarge, CustomAmiId=ami-345678}']
```

```
{
    "ClusterId": "j-123456",
    ...
}

aws emr add-instance-fleet --cluster-id j-123456 --instance-fleet
InstanceFleetType=TASK,TargetSpotCapacity=1,InstanceTypeConfigs=['{InstanceType=m5.xlarge,CustomAmiId=a
{InstanceType=m6g.xlarge, CustomAmiId=ami-345678}']
```

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. By default, Amazon Linux kernel updates and other software packages that require a reboot are not updated. Note that your networking configuration must allow for HTTP and HTTPS egress to Amazon Linux repositories in Amazon S3, otherwise security updates will not succeed.

> **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 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.

> **Note**
> Beginning with Amazon EMR version 5.24.0, you can use a security configuration option to encrypt EBS root device and storage volumes when you specify AWS KMS as your key provider. For more information, see Local disk encryption (p. 376).

You can use an external key provider or an AWS KMS key 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 simplest 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": [
      "*"
      ]
   }
]
}
```

Tutorial: Creating a custom AMI with an encrypted root device volume using a KMS key

The first step in this example is to find the ARN of a KMS key 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 KMS key 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-
xxxxxxxxxxxx
```

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 Use a single custom AMI in an EMR cluster (p. 183). 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. 181).

Best practices and considerations

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

- Amazon EMR 5.30.0 and later, and the Amazon EMR 6.x series are based on Amazon Linux 2. For these Amazon EMR versions, you need to use images based on Amazon Linux 2 for custom AMIs. To find a base custom AMI, see Finding a Linux AMI.
- For Amazon EMR versions earlier than 5.30.0 and 6.x, Amazon Linux 2 AMIs are not supported.
- You must use a 64-bit Amazon Linux AMI. A 32-bit AMI is not 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. 197).
- 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
```

- In cases where you use a VPC with a non-standard domain name and AmazonProvidedDNS, you should not use the `rotate` option in the Operating Systems DNS configuration.

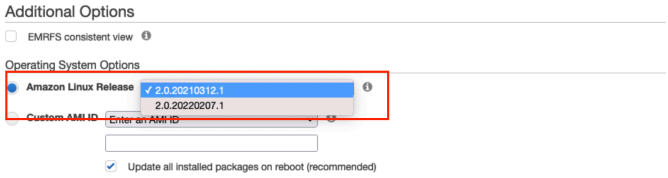
For more information, see Creating an Amazon EBS-backed Linux AMI in the *Amazon EC2 User Guide for Linux Instances*.

Changing the Amazon Linux release for a launching cluster

When you launch a cluster using Amazon EMR 6.6.0 or later, it automatically uses the latest Amazon Linux2 release that has been validated for the default Amazon EMR AMI. You can specify a different Amazon Linux release for your cluster using the Amazon EMR console or the AWS CLI.

To change the Amazon Linux release for a launching cluster (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-6.6.0** 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**, select the **Amazon Linux Release**, and select the Amazon Linux Release version for your cluster.



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

To change the Amazon Linux release for a launching cluster (AWS CLI)

- Use the `--os-release-label` parameter to specify the **Amazon Linux Release** when you run the `aws emr` create-cluster command.

```
aws emr create-cluster --name "Cluster with Different Amazon Linux Release" \
--os-release-label 2.0.20210312.1 \
--release-label emr-6.6.0 --use-default-roles \
--instance-count 2 --instance-type m5.xlarge
```

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.

For more information about Amazon EBS, see Amazon EC2 root device volume.

> **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. However, the minimum root volume size for a custom AMI is also 10 GiB. For more information, see Using a custom AMI (p. 181).

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:

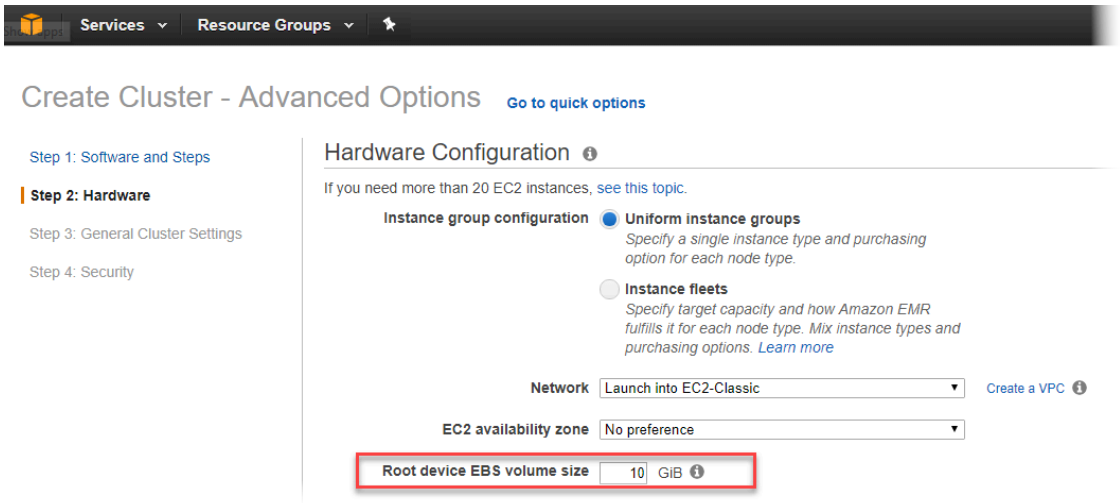
($EBS GB/month) * 0.931323 / 30 / 24 * EMR_EBSRootGiB * InstanceCount

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/emr.
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=m5.xlarge
 InstanceGroupType=CORE,InstanceCount=2,InstanceType=m5.xlarge
```

Configure cluster software

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

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

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

> **Note**
> Bootstrap actions are used differently in Amazon EMR release 4.x and later. For more information about these differences from Amazon EMR AMI versions 2.x and 3.x, go to Differences introduced in 4.x in the Amazon EMR Release Guide.

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 Customizing cluster and application configuration with earlier AMI versions of Amazon EMR in the Amazon EMR Release Guide.

Topics

- Bootstrap action basics (p. 191)
- Conditionally run a bootstrap action (p. 192)
- Shutdown actions (p. 192)
- Use custom bootstrap actions (p. 192)

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.

Conditionally run a bootstrap action

In order to only run a bootstrap actions on the master node, you can use a custom bootstrap action with some logic to determine if the node is master.

```
#!/bin/bash
if grep isMaster /mnt/var/lib/info/instance.json | grep false;
then
    echo "This is not master node, do nothing,exiting"
    exit 0
fi
echo "This is master, continuing to execute script"
# continue with code logic for master node below
```

The following output will print from a core node.

```
This is not master node, do nothing, exiting
```

The following output will print from master node.

```
This is master, continuing to execute script
```

To use this logic, upload your bootstrap action, including the above code, to your Amazon S3 bucket. On the AWS CLI, add the `--bootstrap-actions` parameter to the `aws emr create-cluster` API call and specify your bootstrap script location as the value of `Path`.

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. 593).

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.

Note
For the best performance, we recommend that you store custom bootstrap actions, scripts, and other files that you want to use with Amazon EMR in an Amazon S3 bucket that is in the same AWS Region as your cluster.

Contents

- Add custom bootstrap actions using the AWS CLI or the Amazon EMR CLI (p. 193)
- Add custom bootstrap actions using the console (p. 194)
- Use a custom bootstrap action to copy an object from Amazon S3 to each node (p. 194)

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 https://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, replacing *myKey* with the name of your EC2 key pair. Include `--bootstrap-actions` as a parameter and specify your bootstrap script location as the value of `Path`.

  - 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 m5.xlarge \
    --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 m5.xlarge --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/emr.
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. 564)

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.

```
#!/bin/bash
    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-5.36.0 \
--use-default-roles --ec2-attributes KeyName=myKey \
--applications Name=Hive Name=Pig \
--instance-count 3 --instance-type m5.xlarge \
--bootstrap-actions Path="s3://mybucket/myscriptfolder/copymyfile.sh"
```

Configure cluster hardware and networking

An important consideration when you create an Amazon EMR cluster is how you configure Amazon EC2 instances and network options. This chapter covers the following options, and then ties them all together with best practices and guidelines (p. 338).

- **Node types** – Amazon EC2 instances in an EMR cluster are organized into *node types*. There are three: *master nodes*, *core nodes*, 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. For more information, see Understand node types: master, core, and task nodes (p. 195).
- **EC2 instances** – When you create a cluster, you make choices about the Amazon EC2 instances that each type of node will run on. The EC2 instance type determines the processing and storage profile of the node. The choice of Amazon EC2 instance for your nodes is important because it determines the performance profile of individual node types in your cluster. For more information, see Configure Amazon EC2 instances (p. 197).
- **Networking** – You can launch your Amazon EMR cluster into a VPC using a public subnet, private subnet, or a shared subnet. Your networking configuration determines how customers and services can connect to clusters to perform work, how clusters connect to data stores and other AWS resources, and the options you have for controlling traffic on those connections. For more information, see Configure networking (p. 306).
- **Instance grouping** – The collection of EC2 instances that host each node type is called either an *instance fleet* or a *uniform instance group*. The instance grouping configuration is a choice you make when you create a cluster. This choice determines how you can add nodes to your cluster while it is running. The configuration applies to all node types. It can't be changed later. For more information, see Create a cluster with instance fleets or uniform instance groups (p. 315).

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

Understand node types: master, core, 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. It also runs the HDFS NameNode service, tracks the status of jobs submitted to the cluster, and monitors the health of the instance groups.

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. 593). 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. 564). 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. 599).

Note
With Amazon EMR 5.23.0 and later, you can launch a cluster with three master nodes to support high availability of applications like YARN Resource Manager, HDFS NameNode, Spark, Hive, and Ganglia. The master node is no longer a potential single point of failure with this feature. If

one of the master nodes fails, Amazon EMR automatically fails over to a standby master node and replaces the failed master node with a new one with the same configuration and bootstrap actions. For more information, see Plan and Configure Master Nodes.

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.

There is only one core instance group or instance fleet per cluster, but there can be multiple nodes running on multiple Amazon EC2 instances in the instance group or instance fleet. With instance groups, you can add and remove Amazon EC2 instances while the cluster is running. You can also set up automatic scaling to add instances based on the value of a metric. For more information about adding and removing Amazon EC2 instances with the instance groups configuration, see Scaling cluster resources (p. 610).

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. 317).

> **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. 339).

Task nodes

You can use task nodes 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 Amazon EC2 instances to an existing uniform instance group or by modifying target capacities for a task instance fleet.

With the uniform instance group configuration, you can have up to a total of 48 task instance groups. The ability to add instance groups in this way allows you to mix Amazon 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.

With the instance fleet configuration, 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 do not 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 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.

Beginning with Amazon EMR 6.x release series, the YARN node labels feature is disabled by default. The application master processes can run on both core and task nodes by default. You can enable the YARN node labels feature by configuring following properties:

- `yarn.node-labels.enabled: true`

- `yarn.node-labels.am.default-node-label-expression: 'CORE'`

For information about specific properties, see Amazon EMR settings to prevent job failure because of task node Spot Instance termination (p. 339).

Configure Amazon EC2 instances

EC2 instances come in different configurations known as *instance types*. Instance types have different CPU, input/output, and storage capacities. In addition to the instance type, you can choose different purchasing options for Amazon EC2 instances. You can specify different instance types and purchasing options within uniform instance groups or instance fleets. For more information, see Create a cluster with instance fleets or uniform instance groups (p. 315). For guidance about choosing instance types and purchasing options for your application, see Cluster configuration guidelines and best practices (p. 338).

> **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. 197)
- Instance purchasing options (p. 302)
- Instance storage (p. 303)

Supported instance types

This section describes the instance types that Amazon EMR supports, organized by AWS Region. To learn more about instance types, see Amazon EC2 instances and Amazon Linux AMI instance type matrix.

Not all instance types are available in all Regions, and instance availability is subject to availability and demand in the specified Region and Availability Zone. An instance's Availability Zone is determined by the subnet you use to launch your cluster.

Considerations

Consider the following when you choose instance types for your Amazon EMR cluster.

> **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.

- If you create a cluster using an instance type that is not available in the specified Region and Availability Zone, your cluster may fail to provision or may be stuck provisioning. For information about instance availability, see the Amazon EMR pricing page or see the Supported instance types by AWS Region (p. 198) tables on this page.
- 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. For more information, see Linux AMI virtualization types.
- Some instance types support enhanced networking. For more information, see Enhanced Networking on Linux.
- NVIDIA and CUDA drivers are installed on GPU instance types by default.

Supported instance types by AWS Region

The following tables describe the instance types that Amazon EMR supports, organized by AWS Region. The tables also list the earliest Amazon EMR releases in the 4.x, 5.x, or 6.x series that support each instance type. For example, EMR supports m4.16xlarge instances in US East (N. Virginia) in versions 4.8.3 and later, 5.2.1 and later, and 6.0.0 and later.

US East (N. Virginia) - us-east-1

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m4.16xlarge	emr-4.8.3, emr-5.2.1, emr-6.0.0
	m5.xlarge	emr-5.13.0, emr-6.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m6g.8xlarge	emr-5.30.0, emr-6.1.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	c5d.2xlarge	emr-5.13.0, emr-6.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0
	c7g.xlarge	emr-6.7.0
	c7g.2xlarge	emr-6.7.0
	c7g.4xlarge	emr-6.7.0
	c7g.8xlarge	emr-6.7.0
	c7g.12xlarge	emr-6.7.0
	c7g.16xlarge	emr-6.7.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0

Instance class	Instance type	Minimum supported Amazon EMR version
	c6i.32xlarge	emr-5.35.0, emr-6.6.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0
	c5ad.xlarge	emr-5.33.0, emr-6.3.0
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0
GPU instances	p2.xlarge	emr-5.10.0, emr-6.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	p3.2xlarge	emr-5.10.0, emr-6.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0
	g3.4xlarge	emr-5.18.0, emr-6.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0
Memory optimized	z1d.xlarge	emr-5.17.0, emr-6.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0
	r5.xlarge	emr-5.13.0, emr-6.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r5d.16xlarge	emr-5.13.0, emr-6.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0
	r6a.xlarge	emr-6.8.0
	r6a.2xlarge	emr-6.8.0
	r6a.4xlarge	emr-6.8.0
	r6a.8xlarge	emr-6.8.0
	r6a.12xlarge	emr-6.8.0
	r6a.16xlarge	emr-6.8.0
	r6a.24xlarge	emr-6.8.0
	r6a.32xlarge	emr-6.8.0
	r6a.48xlarge	emr-6.8.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r6i.24xlarge	emr-5.35.0, emr-6.6.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	i3en.6xlarge	emr-5.25.0, emr-6.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0
	i4i.xlarge	emr-6.8.0
	i4i.2xlarge	emr-6.8.0
	i4i.4xlarge	emr-6.8.0
	i4i.8xlarge	emr-6.8.0
	i4i.16xlarge	emr-6.8.0
	i4i.32xlarge	emr-6.8.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0
	h1.2xlarge	emr-5.17.0, emr-6.0.0
	h1.4xlarge	emr-5.17.0, emr-6.0.0
	h1.8xlarge	emr-5.17.0, emr-6.0.0
	h1.16xlarge	emr-5.17.0, emr-6.0.0
	d3.xlarge	emr-5.33.0, emr-6.3.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0
	d3en.xlarge	emr-5.33.0, emr-6.3.0
	d3en.2xlarge	emr-5.33.0, emr-6.3.0
	d3en.4xlarge	emr-5.33.0, emr-6.3.0
	d3en.6xlarge	emr-5.33.0, emr-6.3.0

Instance class	Instance type	Minimum supported Amazon EMR version
	d3en.8xlarge	emr-5.33.0, emr-6.3.0
	d3en.12xlarge	emr-5.33.0, emr-6.3.0

US East (Ohio) - us-east-2

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m4.16xlarge	emr-4.8.3, emr-5.2.1, emr-6.0.0
	m5.xlarge	emr-5.13.0, emr-6.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m6g.8xlarge	emr-5.30.0, emr-6.1.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	c5d.2xlarge	emr-5.13.0, emr-6.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0
	c7g.xlarge	emr-6.7.0
	c7g.2xlarge	emr-6.7.0
	c7g.4xlarge	emr-6.7.0
	c7g.8xlarge	emr-6.7.0
	c7g.12xlarge	emr-6.7.0
	c7g.16xlarge	emr-6.7.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0

Instance class	Instance type	Minimum supported Amazon EMR version
	c6gd.xlarge	emr-5.33.0, emr-6.3.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0
GPU instances	p2.xlarge	emr-5.10.0, emr-6.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0
	g3.4xlarge	emr-5.18.0, emr-6.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	g4dn.xlarge	emr-5.30.0, emr-6.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0
Memory optimized	z1d.xlarge	emr-5.17.0, emr-6.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0
	r5.xlarge	emr-5.13.0, emr-6.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r5a.16xlarge	emr-5.20.0, emr-6.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0
	r6a.xlarge	emr-6.8.0
	r6a.2xlarge	emr-6.8.0
	r6a.4xlarge	emr-6.8.0
	r6a.8xlarge	emr-6.8.0
	r6a.12xlarge	emr-6.8.0
	r6a.16xlarge	emr-6.8.0
	r6a.24xlarge	emr-6.8.0
	r6a.32xlarge	emr-6.8.0
	r6a.48xlarge	emr-6.8.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r5b.24xlarge	emr-5.33.0, emr-6.3.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0
	i4i.xlarge	emr-6.8.0
	i4i.2xlarge	emr-6.8.0
	i4i.4xlarge	emr-6.8.0
	i4i.8xlarge	emr-6.8.0

Instance class	Instance type	Minimum supported Amazon EMR version
	i4i.16xlarge	emr-6.8.0
	i4i.32xlarge	emr-6.8.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0
	h1.2xlarge	emr-5.17.0, emr-6.0.0
	h1.4xlarge	emr-5.17.0, emr-6.0.0
	h1.8xlarge	emr-5.17.0, emr-6.0.0
	h1.16xlarge	emr-5.17.0, emr-6.0.0
	d3.xlarge	emr-5.33.0, emr-6.3.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0
	d3en.xlarge	emr-5.33.0, emr-6.3.0
	d3en.2xlarge	emr-5.33.0, emr-6.3.0
	d3en.4xlarge	emr-5.33.0, emr-6.3.0
	d3en.6xlarge	emr-5.33.0, emr-6.3.0
	d3en.8xlarge	emr-5.33.0, emr-6.3.0
	d3en.12xlarge	emr-5.33.0, emr-6.3.0

US West (N. California) - us-west-1

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m4.16xlarge	emr-4.8.3, emr-5.2.1, emr-6.0.0
	m5.xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m5.2xlarge	emr-5.13.0, emr-6.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m6i.12xlarge	emr-5.35.0, emr-6.6.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0

Instance class	Instance type	Minimum supported Amazon EMR version
	c6g.2xlarge	emr-5.31.0, emr-6.1.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0
GPU instances	g3.4xlarge	emr-5.18.0, emr-6.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0
Memory optimized	z1d.xlarge	emr-5.17.0, emr-6.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	z1d.3xlarge	emr-5.17.0, emr-6.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0
	r5.xlarge	emr-5.13.0, emr-6.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r6gd.xlarge	emr-5.33.0, emr-6.3.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0
	i4i.xlarge	emr-6.8.0
	i4i.2xlarge	emr-6.8.0
	i4i.4xlarge	emr-6.8.0
	i4i.8xlarge	emr-6.8.0
	i4i.16xlarge	emr-6.8.0

Instance class	Instance type	Minimum supported Amazon EMR version
	i4i.32xlarge	emr-6.8.0

US West (Oregon) - us-west-2

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m4.16xlarge	emr-4.8.3, emr-5.2.1, emr-6.0.0
	m5.xlarge	emr-5.13.0, emr-6.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m6g.12xlarge	emr-5.30.0, emr-6.1.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	c5d.4xlarge	emr-5.13.0, emr-6.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0
	c7g.xlarge	emr-6.7.0
	c7g.2xlarge	emr-6.7.0
	c7g.4xlarge	emr-6.7.0
	c7g.8xlarge	emr-6.7.0
	c7g.12xlarge	emr-6.7.0
	c7g.16xlarge	emr-6.7.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0
	c5ad.xlarge	emr-5.33.0, emr-6.3.0

Instance class	Instance type	Minimum supported Amazon EMR version
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0
GPU instances	p2.xlarge	emr-5.10.0, emr-6.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	p3.8xlarge	emr-5.10.0, emr-6.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0
	g3.4xlarge	emr-5.18.0, emr-6.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0
Memory optimized	z1d.xlarge	emr-5.17.0, emr-6.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0
	r5.xlarge	emr-5.13.0, emr-6.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0
	r6a.xlarge	emr-6.8.0
	r6a.2xlarge	emr-6.8.0
	r6a.4xlarge	emr-6.8.0
	r6a.8xlarge	emr-6.8.0
	r6a.12xlarge	emr-6.8.0
	r6a.16xlarge	emr-6.8.0
	r6a.24xlarge	emr-6.8.0
	r6a.32xlarge	emr-6.8.0
	r6a.48xlarge	emr-6.8.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r5b.xlarge	emr-5.33.0, emr-6.3.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	i3en.12xlarge	emr-5.25.0, emr-6.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0
	i4i.xlarge	emr-6.8.0
	i4i.2xlarge	emr-6.8.0
	i4i.4xlarge	emr-6.8.0
	i4i.8xlarge	emr-6.8.0
	i4i.16xlarge	emr-6.8.0
	i4i.32xlarge	emr-6.8.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0
	h1.2xlarge	emr-5.17.0, emr-6.0.0
	h1.4xlarge	emr-5.17.0, emr-6.0.0
	h1.8xlarge	emr-5.17.0, emr-6.0.0
	h1.16xlarge	emr-5.17.0, emr-6.0.0
	d3.xlarge	emr-5.33.0, emr-6.3.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0
	d3en.xlarge	emr-5.33.0, emr-6.3.0
	d3en.2xlarge	emr-5.33.0, emr-6.3.0
	d3en.4xlarge	emr-5.33.0, emr-6.3.0
	d3en.6xlarge	emr-5.33.0, emr-6.3.0
	d3en.8xlarge	emr-5.33.0, emr-6.3.0

Instance class	Instance type	Minimum supported Amazon EMR version
	d3en.12xlarge	emr-5.33.0, emr-6.3.0

Africa (Cape Town) - af-south-1

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m5.xlarge	emr-5.13.0, emr-6.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
Memory optimized	r5.xlarge	emr-5.13.0, emr-6.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0

Asia Pacific (Hong Kong) - ap-east-1

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m5.xlarge	emr-5.13.0, emr-6.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0

Instance class	Instance type	Minimum supported Amazon EMR version
	c6g.4xlarge	emr-5.31.0, emr-6.1.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0
GPU instances	g4dn.xlarge	emr-5.30.0, emr-6.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0
Memory optimized	r5.xlarge	emr-5.13.0, emr-6.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0

Instance class	Instance type	Minimum supported Amazon EMR version
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0

Asia Pacific (Jakarta) - ap-southeast-3

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m5.xlarge	emr-5.13.0, emr-6.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	c5.2xlarge	emr-5.13.0, emr-6.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0
Memory optimized	r5.xlarge	emr-5.13.0, emr-6.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r5a.16xlarge	emr-5.20.0, emr-6.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0

Asia Pacific (Mumbai) - ap-south-1

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m4.16xlarge	emr-4.8.3, emr-5.2.1, emr-6.0.0
	m5.xlarge	emr-5.13.0, emr-6.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	c5n.18xlarge	emr-5.20.0, emr-6.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0
GPU instances	p2.xlarge	emr-5.10.0, emr-6.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0
Memory optimized	z1d.xlarge	emr-5.17.0, emr-6.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0
	r5.xlarge	emr-5.13.0, emr-6.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0
	r6a.xlarge	emr-6.8.0
	r6a.2xlarge	emr-6.8.0
	r6a.4xlarge	emr-6.8.0
	r6a.8xlarge	emr-6.8.0
	r6a.12xlarge	emr-6.8.0
	r6a.16xlarge	emr-6.8.0
	r6a.24xlarge	emr-6.8.0
	r6a.32xlarge	emr-6.8.0
	r6a.48xlarge	emr-6.8.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	i3.8xlarge	emr-5.9.0, emr-6.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0

Asia Pacific (Osaka) - ap-northeast-3

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m4.16xlarge	emr-4.8.3, emr-5.2.1, emr-6.0.0
	m5.xlarge	emr-5.13.0, emr-6.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
Memory optimized	r5.xlarge	emr-5.13.0, emr-6.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0

Asia Pacific (Seoul) - ap-northeast-2

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m4.16xlarge	emr-4.8.3, emr-5.2.1, emr-6.0.0
	m5.xlarge	emr-5.13.0, emr-6.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0
GPU instances	p2.xlarge	emr-5.10.0, emr-6.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0
Memory optimized	z1d.xlarge	emr-5.17.0, emr-6.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	z1d.3xlarge	emr-5.17.0, emr-6.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0
	r5.xlarge	emr-5.13.0, emr-6.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0

Instance class	Instance type	Minimum supported Amazon EMR version
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0

Asia Pacific (Singapore) - ap-southeast-1

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m4.16xlarge	emr-4.8.3, emr-5.2.1, emr-6.0.0
	m5.xlarge	emr-5.13.0, emr-6.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m5a.2xlarge	emr-5.20.0, emr-6.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	c5.12xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0
	c5ad.xlarge	emr-5.33.0, emr-6.3.0
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0

Instance class	Instance type	Minimum supported Amazon EMR version
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0
GPU instances	p2.xlarge	emr-5.10.0, emr-6.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0
	g3.4xlarge	emr-5.18.0, emr-6.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0
Memory optimized	z1d.xlarge	emr-5.17.0, emr-6.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0
	r5.xlarge	emr-5.13.0, emr-6.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r6g.xlarge	emr-5.31.0, emr-6.1.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r6i.12xlarge	emr-5.35.0, emr-6.6.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0
	i4i.xlarge	emr-6.8.0
	i4i.2xlarge	emr-6.8.0
	i4i.4xlarge	emr-6.8.0
	i4i.8xlarge	emr-6.8.0
	i4i.16xlarge	emr-6.8.0
	i4i.32xlarge	emr-6.8.0

Asia Pacific (Sydney) - ap-southeast-2

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m4.16xlarge	emr-4.8.3, emr-5.2.1, emr-6.0.0
	m5.xlarge	emr-5.13.0, emr-6.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m5.16xlarge	emr-5.13.0, emr-6.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	c5.4xlarge	emr-5.13.0, emr-6.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0

Instance class	Instance type	Minimum supported Amazon EMR version
	c5ad.xlarge	emr-5.33.0, emr-6.3.0
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0
GPU instances	p2.xlarge	emr-5.10.0, emr-6.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0
	g3.4xlarge	emr-5.18.0, emr-6.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0
Memory optimized	z1d.xlarge	emr-5.17.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	z1d.2xlarge	emr-5.17.0, emr-6.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0
	r5.xlarge	emr-5.13.0, emr-6.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r6g.16xlarge	emr-5.31.0, emr-6.1.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0

Asia Pacific (Tokyo) - ap-northeast-1

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m4.16xlarge	emr-4.8.3, emr-5.2.1, emr-6.0.0
	m5.xlarge	emr-5.13.0, emr-6.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0

Instance class	Instance type	Minimum supported Amazon EMR version
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0
GPU instances	p2.xlarge	emr-5.10.0, emr-6.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0
	g3.4xlarge	emr-5.18.0, emr-6.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0
Memory optimized	z1d.xlarge	emr-5.17.0, emr-6.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	z1d.3xlarge	emr-5.17.0, emr-6.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0
	r5.xlarge	emr-5.13.0, emr-6.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r5dn.xlarge	emr-5.31.0, emr-6.1.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	i3.4xlarge	emr-5.9.0, emr-6.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0
	i4i.xlarge	emr-6.8.0
	i4i.2xlarge	emr-6.8.0
	i4i.4xlarge	emr-6.8.0
	i4i.8xlarge	emr-6.8.0
	i4i.16xlarge	emr-6.8.0
	i4i.32xlarge	emr-6.8.0

Canada (Central) - ca-central-1

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m4.16xlarge	emr-4.8.3, emr-5.2.1, emr-6.0.0
	m5.xlarge	emr-5.13.0, emr-6.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	c5n.xlarge	emr-5.20.0, emr-6.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0
GPU instances	p3.2xlarge	emr-5.10.0, emr-6.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0
	g3.4xlarge	emr-5.18.0, emr-6.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0
Memory optimized	r5.xlarge	emr-5.13.0, emr-6.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0

Europe (Frankfurt) - eu-central-1

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m4.16xlarge	emr-4.8.3, emr-5.2.1, emr-6.0.0
	m5.xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m5.2xlarge	emr-5.13.0, emr-6.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m6i.12xlarge	emr-5.35.0, emr-6.6.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	c5n.2xlarge	emr-5.20.0, emr-6.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0
	c5ad.xlarge	emr-5.33.0, emr-6.3.0
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0

Instance class	Instance type	Minimum supported Amazon EMR version
	c6i.xlarge	emr-5.35.0, emr-6.6.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0
GPU instances	p2.xlarge	emr-5.10.0, emr-6.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0
	g3.4xlarge	emr-5.18.0, emr-6.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0
Memory optimized	z1d.xlarge	emr-5.17.0, emr-6.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0
	r5.xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0
	r6a.xlarge	emr-6.8.0
	r6a.2xlarge	emr-6.8.0
	r6a.4xlarge	emr-6.8.0
	r6a.8xlarge	emr-6.8.0
	r6a.12xlarge	emr-6.8.0
	r6a.16xlarge	emr-6.8.0
	r6a.24xlarge	emr-6.8.0
	r6a.32xlarge	emr-6.8.0
	r6a.48xlarge	emr-6.8.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r6g.2xlarge	emr-5.31.0, emr-6.1.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r6i.16xlarge	emr-5.35.0, emr-6.6.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0
	i4i.xlarge	emr-6.8.0
	i4i.2xlarge	emr-6.8.0
	i4i.4xlarge	emr-6.8.0
	i4i.8xlarge	emr-6.8.0
	i4i.16xlarge	emr-6.8.0
	i4i.32xlarge	emr-6.8.0

Europe (Ireland) - eu-west-1

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m4.16xlarge	emr-4.8.3, emr-5.2.1, emr-6.0.0
	m5.xlarge	emr-5.13.0, emr-6.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m5.24xlarge	emr-5.13.0, emr-6.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0

Instance class	Instance type	Minimum supported Amazon EMR version
	c6g.2xlarge	emr-5.31.0, emr-6.1.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0
	c7g.xlarge	emr-6.7.0
	c7g.2xlarge	emr-6.7.0
	c7g.4xlarge	emr-6.7.0
	c7g.8xlarge	emr-6.7.0
	c7g.12xlarge	emr-6.7.0
	c7g.16xlarge	emr-6.7.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0
	c5ad.xlarge	emr-5.33.0, emr-6.3.0
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0

Instance class	Instance type	Minimum supported Amazon EMR version
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0
GPU instances	p2.xlarge	emr-5.10.0, emr-6.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0
	g3.4xlarge	emr-5.18.0, emr-6.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0
Memory optimized	z1d.xlarge	emr-5.17.0, emr-6.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0
	r5.xlarge	emr-5.13.0, emr-6.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r6g.8xlarge	emr-5.31.0, emr-6.1.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	i3en.2xlarge	emr-5.25.0, emr-6.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0
	i4i.xlarge	emr-6.8.0
	i4i.2xlarge	emr-6.8.0
	i4i.4xlarge	emr-6.8.0
	i4i.8xlarge	emr-6.8.0
	i4i.16xlarge	emr-6.8.0
	i4i.32xlarge	emr-6.8.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0
	h1.2xlarge	emr-5.17.0, emr-6.0.0
	h1.4xlarge	emr-5.17.0, emr-6.0.0
	h1.8xlarge	emr-5.17.0, emr-6.0.0
	h1.16xlarge	emr-5.17.0, emr-6.0.0
	d3.xlarge	emr-5.33.0, emr-6.3.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0
	d3en.xlarge	emr-5.33.0, emr-6.3.0
	d3en.2xlarge	emr-5.33.0, emr-6.3.0

Instance class	Instance type	Minimum supported Amazon EMR version
	d3en.4xlarge	emr-5.33.0, emr-6.3.0
	d3en.6xlarge	emr-5.33.0, emr-6.3.0
	d3en.8xlarge	emr-5.33.0, emr-6.3.0
	d3en.12xlarge	emr-5.33.0, emr-6.3.0

Europe (London) - eu-west-2

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m4.16xlarge	emr-4.8.3, emr-5.2.1, emr-6.0.0
	m5.xlarge	emr-5.13.0, emr-6.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m6g.2xlarge	emr-5.30.0, emr-6.1.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0
GPU instances	p3.2xlarge	emr-5.10.0, emr-6.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0
Memory optimized	z1d.xlarge	emr-5.17.0, emr-6.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0
	r5.xlarge	emr-5.13.0, emr-6.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r5a.24xlarge	emr-5.20.0, emr-6.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0

Europe (Milan) - eu-south-1

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m5.xlarge	emr-5.13.0, emr-6.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0
Memory optimized	r5.xlarge	emr-5.13.0, emr-6.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r5.24xlarge	emr-5.13.0, emr-6.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0

Europe (Paris) - eu-west-3

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m5.xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m5.2xlarge	emr-5.13.0, emr-6.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0
GPU instances	g4dn.xlarge	emr-5.30.0, emr-6.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0
Memory optimized	r5.xlarge	emr-5.13.0, emr-6.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	i3.2xlarge	emr-5.9.0, emr-6.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0

Europe (Stockholm) - eu-north-1

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m5.xlarge	emr-5.13.0, emr-6.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	c5.9xlarge	emr-5.13.0, emr-6.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0
GPU instances	g4dn.xlarge	emr-5.30.0, emr-6.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0
Memory optimized	r5.xlarge	emr-5.13.0, emr-6.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0

Middle East (Bahrain) - me-south-1

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m5.xlarge	emr-5.13.0, emr-6.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0
GPU instances	g4dn.xlarge	emr-5.30.0, emr-6.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0
Memory optimized	r5.xlarge	emr-5.13.0, emr-6.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0

Middle East (UAE) - me-central-1

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m5.xlarge	emr-5.13.0, emr-6.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m6g.8xlarge	emr-5.30.0, emr-6.1.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0
Memory optimized	r5.xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0

South America (São Paulo) - sa-east-1

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m4.16xlarge	emr-4.8.3, emr-5.2.1, emr-6.0.0
	m5.xlarge	emr-5.13.0, emr-6.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0
GPU instances	g4dn.xlarge	emr-5.30.0, emr-6.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
Memory optimized	r5.xlarge	emr-5.13.0, emr-6.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	i3.8xlarge	emr-5.9.0, emr-6.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0

China (Ningxia) - cn-northwest-1

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m4.16xlarge	emr-4.8.3, emr-5.2.1, emr-6.0.0
	m5.xlarge	emr-5.13.0, emr-6.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m6g.8xlarge	emr-5.30.0, emr-6.1.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0
GPU instances	p3.2xlarge	emr-5.10.0, emr-6.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0
Memory optimized	z1d.xlarge	emr-5.17.0, emr-6.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0
	r5.xlarge	emr-5.13.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	i3.16xlarge	emr-5.9.0, emr-6.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0

China (Beijing) - cn-north-1

Instance class	Instance type	Minimum supported Amazon EMR version
General purpose	m4.16xlarge	emr-4.8.3, emr-5.2.1, emr-6.0.0
	m5.xlarge	emr-5.13.0, emr-6.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	m5a.16xlarge	emr-5.20.0, emr-6.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0
Compute optimized	c5.xlarge	emr-5.13.0, emr-6.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0
GPU instances	p2.xlarge	emr-5.10.0, emr-6.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0

Instance class	Instance type	Minimum supported Amazon EMR version
	p3.8xlarge	emr-5.10.0, emr-6.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0
	g3.4xlarge	emr-5.18.0, emr-6.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0
Memory optimized	r5.xlarge	emr-5.13.0, emr-6.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0

Instance class	Instance type	Minimum supported Amazon EMR version
	r6g.8xlarge	emr-5.31.0, emr-6.1.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0
Storage optimized	i3.xlarge	emr-5.9.0, emr-6.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0

Previous generation instances

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	m1.small[1] | m1.medium[1] | m1.large[1] | m1.xlarge[1] | m2.xlarge[1] | m2.2xlarge[1] | m2.4xlarge[1] | m3.xlarge[1]| m3.2xlarge[1]
Compute-optimized	c1.medium[1] [2] | c1.xlarge[1] | c3.xlarge[1] | c3.2xlarge[1] | 3.4xlarge[1]| c3.8xlarge[1] | cc2.8xlarge
Memory-optimized	r3.xlarge | r3.2xlarge | r3.4xlarge | r3.8xlarge
Storage-optimized	i2.xlarge | i2.2xlarge | i2.4xlarge | i2.8xlarge
Accelerated computing (GPU)	g2.2xlarge

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

[2] Not supported in release version 5.15.0.

Instance purchasing options

When you set up a cluster, you choose a purchasing option for Amazon EC2 instances. You can choose On-Demand Instances, Spot Instances, or both. Prices vary based on the instance type and Region. The Amazon EMR price is in addition to the Amazon EC2 price (the price for the underlying servers) and Amazon EBS price (if attaching Amazon EBS volumes). 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, you can only specify the purchasing option for an instance group when you create it, and the instance type and purchasing option apply to all Amazon EC2 instances in each instance group. 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. 315).

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 `m5.xlarge` Reserved Instance with the instance reservation scoped to the US-East Region. You then launch an Amazon EMR cluster in US-East that uses two `m5.xlarge` 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 if Spot capacity becomes unavailable for the instance type you are running. For

more information about when using Spot Instances may be appropriate for your application, see When should you use Spot Instances? (p. 339).

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*.

> **Note**
> Spot Instances with a defined duration (also known as Spot blocks) are no longer available to new customers from July 1, 2021. For customers who have previously used the feature, we will continue to support Spot Instances with a defined duration until December 31, 2022.

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. 306).

> **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 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. 315).

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. For more information, see Configure instance fleets (p. 316).

Instance storage

Instance store and Amazon EBS volume storage is used for HDFS data and for buffers, caches, scratch data, and other temporary content that some applications may "spill" to the local file system.

Amazon EBS works differently within Amazon EMR than it does with regular Amazon EC2 instances. Amazon EBS volumes attached to Amazon 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 Amazon 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).

Other caveats for using Amazon EBS with Amazon EMR clusters are:

- You can't snapshot an Amazon 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. 181).
- An encrypted Amazon EBS root device 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. 186).
- 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.
- The Amazon EBS volumes on core nodes cannot be less than 5 GB.

Default Amazon EBS storage for instances

Amazon EMR automatically attaches an Amazon EBS General Purpose SSD (gp2) 10 GB volume as the root device for its AMIs to enhance performance. In addition, for EC2 instances with EBS-only storage, Amazon EMR allocates Amazon EBS gp2 storage volumes to instances. When you create a cluster with Amazon EMR release version 5.22.0 and later, the default amount of Amazon EBS storage increases based on the size of the instance. We split increased storage across multiple volumes, giving increased IOPS performance and, in turn, increased performance for some standardized workloads. If you want to use a different Amazon EBS gp2 instance storage configuration, you can specify this when you create an Amazon EMR cluster or add nodes to an existing cluster. At this time, Amazon EBS gp3 volumes can't be used as root volumes on an Amazon EMR cluster. You can only use Amazon EBS gp2 volumes as root volumes, and add gp3 volumes as additional volumes. The following table identifies the default number of Amazon EBS gp2 storage volumes, sizes, and total sizes per instance type.

Amazon EBS costs are pro-rated by the hour based on the monthly charges for gp2 volumes in the AWS Region where the cluster runs. For example, the Amazon 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).

Default Amazon EBS gp2 storage volumes and size by instance type for Amazon EMR 5.22.0 and later

Instance size	Number of volumes	Volume size (GiB)	Total size (GiB)
*.large	1	32	32
*.xlarge	2	32	64
*.2xlarge	4	32	128
*.4xlarge	4	64	256
*.8xlarge	4	128	512
*.9xlarge	4	144	576
*.10xlarge	4	160	640

Instance size	Number of volumes	Volume size (GiB)	Total size (GiB)
*.12xlarge	4	192	768
*.16xlarge	4	256	1024
*.18xlarge	4	288	1152
*.24xlarge	4	384	1536

Specifying additional EBS storage volumes

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

Amazon EMR doesn't allow you to modify your volume type from gp2 to gp3 for an existing EMR cluster. To use gp3 for your workloads/use-cases, you need to launch a new EMR cluster. Additionally, we do not recommend updating the throughput and IOPS on a cluster that is in use or that is being provisioned, since Amazon EMR uses the throughput and IOPS values you specified at cluster launch time for any new instance added during cluster scale-up. See Comparing Amazon EBS volume types gp2 and gp3 (p. 305) and Selecting IOPS and throughput when migrating to gp3 (p. 306).

> **Important**
> To use a gp3 volume with your EMR cluster, launch a new EMR cluster using the API, SDK or CLI.

Comparing Amazon EBS volume types gp2 and gp3

Here is a comparison of cost between gp2 and gp3 volumes in the us-east-1 (N. Virginia) region

Volume type	gp3	gp2
Volume size	1 GiB – 16 TiB	1 GiB – 16 TiB
Default/Baseline IOPS	3000	3 IOPS/GiB (minimum 100 IOPS) to a maximum of 16,000 IOPS. Volumes smaller than 1 TiB can also burst up to 3,000 IOPS.
Max IOPS/volume	16,000	16,000
Default/Baseline throughput	125 MiB/s	Throughput limit is between 128 MiB/s and 250 MiB/s, depending on the volume size.
Max throughput/volume	1,000 MiB/s	250 MiB/s
Price	$0.08/GiB-month 3,000 IOPS free and $0.005/provisioned	$0.10/GiB-month

Volume type	gp3	gp2
	IOPS-month over 3,000; 125 MiB/s free and $0.04/ provisioned MiB/s-month over 125MiB/s	

Selecting IOPS and throughput when migrating to gp3

When provisioning a gp2 volume, you must figure out the size of the volume in order to get the proportional IOPS and throughput. With gp3, you don't have to provision a bigger volume to get higher performance. You can choose your desired size and performance according to application need. Selecting the right size and right performance parameters (IOPS, throughput) can provide you maximum cost reduction, without affecting performance.

Here is a table to help you select gp3 configuration options:

Volume size	IOPS	Throughput
1–170 GiB	3000	125 MiB/s
170–334 GiB	3000	125 MiB/s if the chosen EC2 instance type supports 125MiB/s or less, use higher as per usage, Max 250 MiB/s*.
334–1000 GiB	3000	125 MiB/s if the chosen EC2 instance type supports 125MiB/s or less, Use higher as per usage, Max 250 MiB/s*.
1000+ GiB	Match gp2 IOPS (Size in GiB x 3) or Max IOPS driven by current gp2 volume	125 MiB/s if the chosen EC2 instance type supports 125MiB/s or less, Use higher as per usage, Max 250 MiB/s*.

*Gp3 has the capability to provide throughput up to 1000 MiB/s. Since gp2 provides a maximum of 250MiB/s throughput, you may not need to go beyond this limit when you use gp3.

Configure networking

Most clusters launch into a virtual network using Amazon Virtual Private Cloud (Amazon VPC). A VPC is an isolated virtual network within AWS that is logically isolated within your AWS account. You can configure aspects such as private IP address ranges, subnets, routing tables, and network gateways. For more information, see the Amazon VPC User Guide.

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 Amazon EMR clusters in both public and private VPC subnets. This means you do not need internet connectivity to run an Amazon 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 release version 4.2 and later.

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

Topics

- Amazon VPC options (p. 307)
- Set up a VPC to host clusters (p. 311)
- Launch clusters into a VPC (p. 312)
- Minimum Amazon S3 policy for private subnet (p. 313)
- More resources for learning about VPCs (p. 314)

Amazon VPC options

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

Public subnets

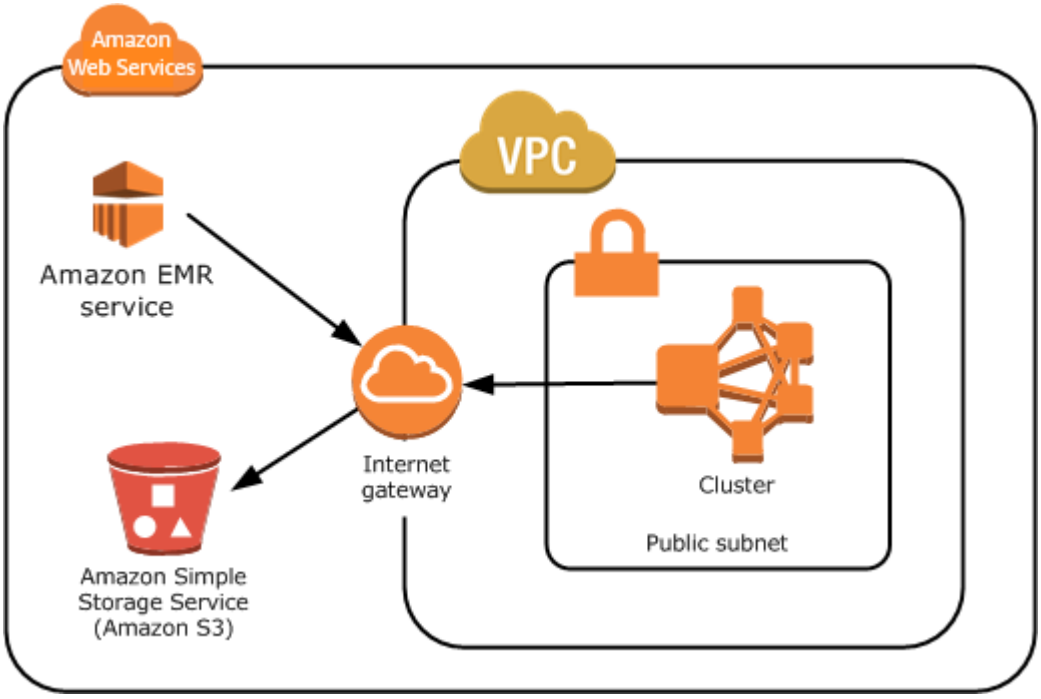
EMR clusters in a public subnet require a connected internet gateway. This is because Amazon EMR clusters must access AWS services and Amazon EMR. If a service, such as Amazon S3, provides the ability to create a VPC endpoint, you can access those services using the endpoint instead of accessing a public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

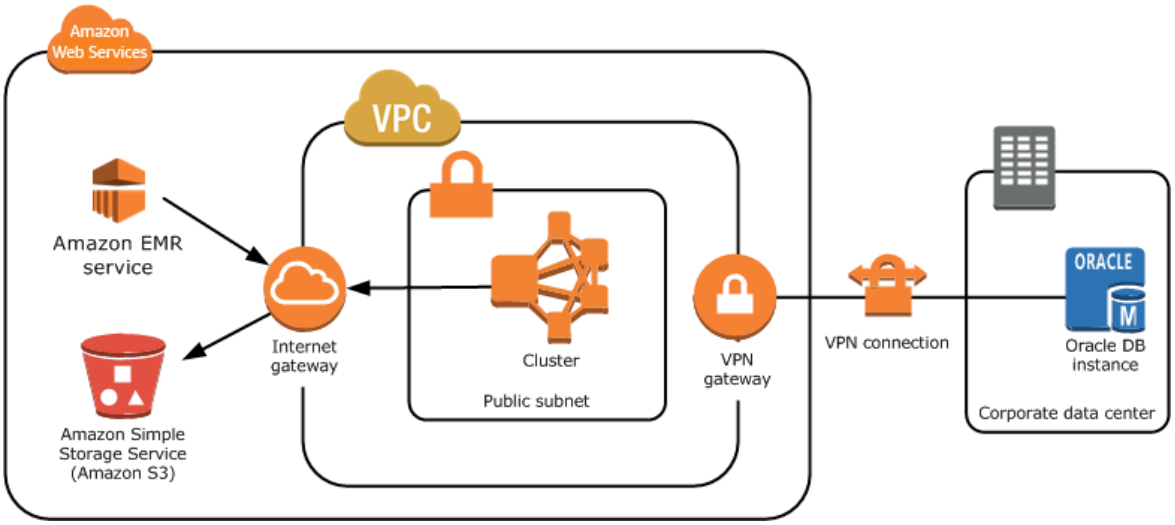
If you have additional AWS resources that you do not want connected to the internet gateway, you can launch those components in a private subnet that you create within your VPC.

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. 532).

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

A private subnet lets you launch AWS resources without requiring the subnet to have an attached internet gateway. Amazon EMR supports launching clusters in private subnets with release versions 4.2.0 or later.

> **Note**
> When you set up an Amazon EMR cluster in a private subnet, we recommend that you also set up VPC endpoints for Amazon S3. If your EMR cluster is in a private subnet without VPC endpoints for Amazon S3, you will incur additional NAT gateway charges that are associated with S3 traffic because the traffic between your EMR cluster and S3 will not stay within your VPC.

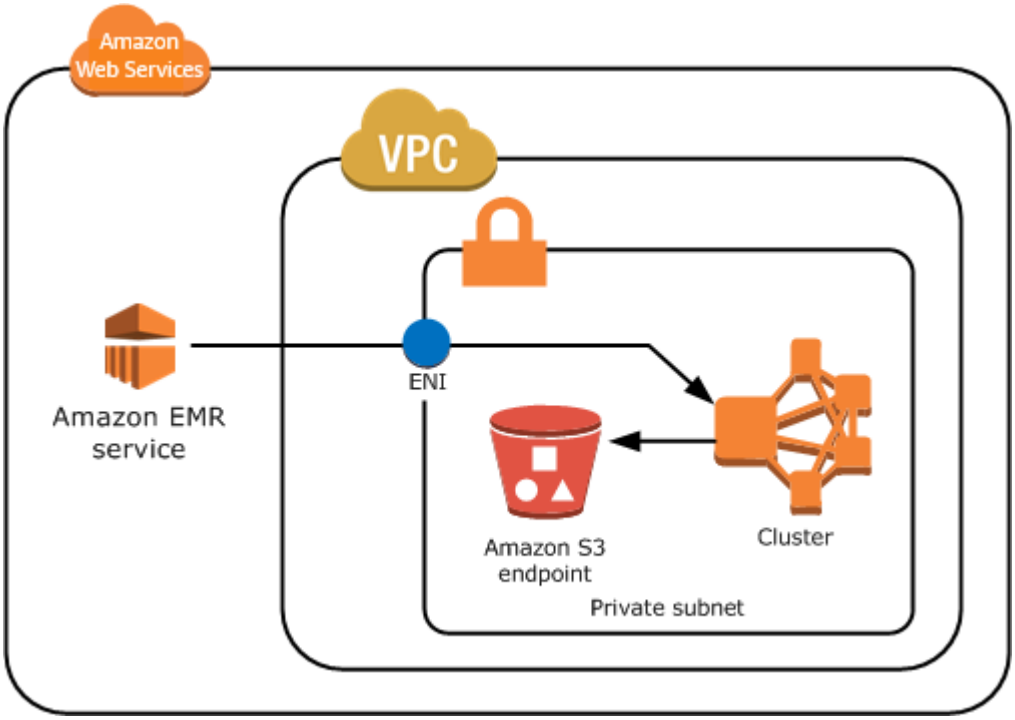
Private subnets differ from public subnets in the following ways:

- 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. 313)
- 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 Amazon 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 Amazon EMR cluster from public to private or vice versa. To locate an Amazon EMR cluster within a private subnet, the cluster must be started in that private subnet.

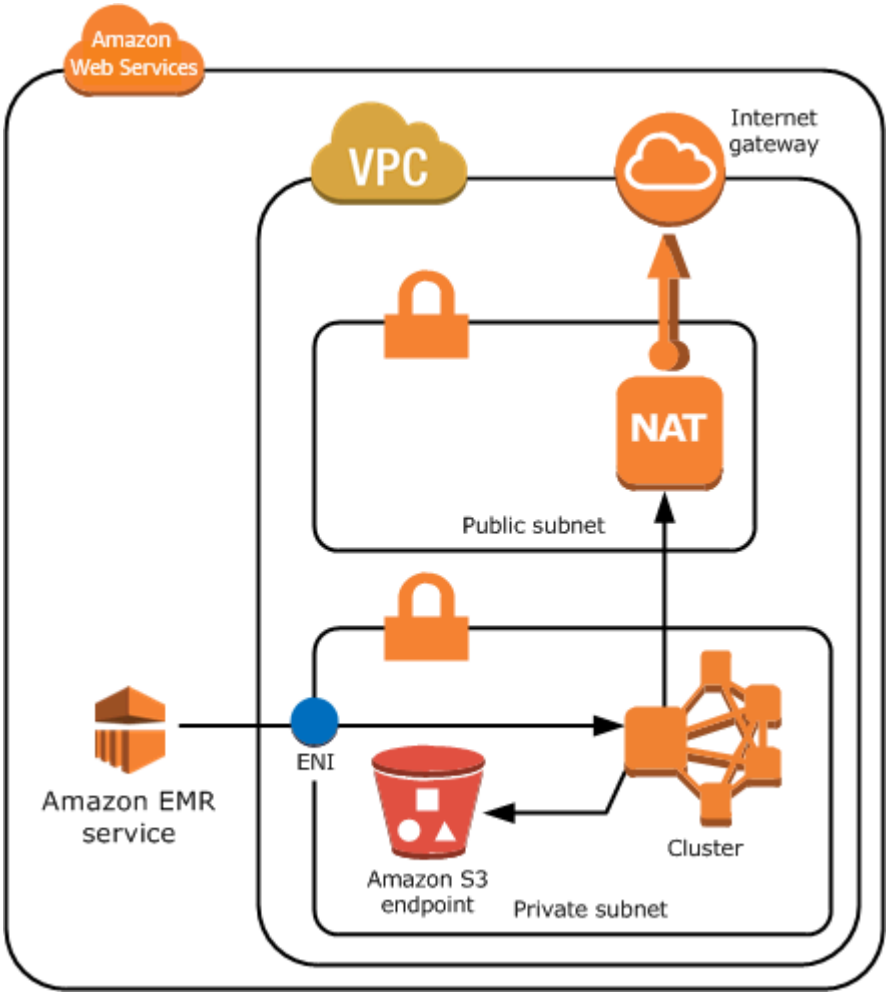
Amazon EMR creates and uses different default security groups for the clusters in a private subnet: ElasticMapReduce-Master-Private, ElasticMapReduce-Slave-Private, and ElasticMapReduce-ServiceAccess. For more information, see Control network traffic with security groups (p. 532).

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 Amazon 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 Amazon EMR cluster within a private subnet connected to a NAT instance that is residing in a public subnet.



Shared subnets

VPC sharing allows customers to share subnets with other AWS accounts within the same AWS Organization. You can launch Amazon EMR clusters into both public shared and private shared subnets, with the following caveats.

The subnet owner must share a subnet with you before you can launch an Amazon EMR cluster into it. However, shared subnets can later be unshared. For more information, see Working with Shared VPCs. When a cluster is launched into a shared subnet and that shared subnet is then unshared, you can observe specific behaviors based on the state of the Amazon EMR cluster when the subnet is unshared.

- Subnet is unshared *before* the cluster is successfully launched - If the owner stops sharing the Amazon VPC or subnet while the participant is launching a cluster, the cluster could fail to start or be partially initialized without provisioning all requested instances.
- Subnet is unshared *after* the cluster is successfully launched - When the owner stops sharing a subnet or Amazon VPC with the participant, the participant's clusters will not be able to resize to add new instances or to replace unhealthy instances.

When you launch an Amazon EMR cluster, multiple security groups are created. In a shared subnet, the subnet participant controls these security groups. The subnet owner can see these security groups but cannot perform any actions on them. If the subnet owner wants to remove or modify the security group, the participant that created the security group must take the action.

Control VPC permissions with IAM

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

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 Amazon 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 example, you can create one user account to administer subnets, and a second user account that can launch clusters but cannot modify Amazon VPC settings. 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*.

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 VPC with subnets for an Amazon EMR cluster

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the top-right of the page, choose the AWS Region for your VPC.
3. Choose **Create VPC**.
4. On the **VPC settings** page, choose **VPC and more**.
5. Under **Name tag auto-generation**, enable **Auto-generate** and enter a name for your VPC. This helps you to identify the VPC and subnet in the Amazon VPC console after you've created them.
6. In the **IPv4 CIDR block** field, enter 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
7. Under **Number of Availability Zones (AZs)**, choose the number of Availability Zones you want to launch your subnets in.
8. Under **Number of public subnets**, choose a single public subnet to add to your VPC. If the data used by the cluster is available on the internet (for example, in Amazon S3 or Amazon RDS), you only need to use a public subnet and don't need to add a private subnet.
9. Under **Number of private subnets**, choose the number of private subnets you want to add to your VPC. Select one or more if the the data for your application is stored in your own network (for example, in an Oracle database). For a VPC in a private subnet, all Amazon 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.
10. Under **NAT gateways**, optionally choose to add NAT gateways. They are only necessary if you have private subnets that need to communicate with the internet.
11. Under **VPC endpoints**, optionally choose to add endpoints for Amazon S3 to your subnets.
12. Verify that **Enable DNS hostnames** and**Enable DNS resolution** are checked. For more information, see Using DNS with your VPC.
13. Choose **Create VPC**.

14. A status window shows the work in progress. When the work completes, choose **View VPC** to navigate to the **Your VPCs** page, which displays your default VPC and the VPC that you just created. The VPC that you created is a nondefault VPC, therefore the **Default VPC** column displays **No**.
15. If you want to associate your VPC with a DNS entry that does not include a domain name, navigate to **DHCP option sets**, choose **Create DHCP options set**, and omit a domain name. After you create your option set, navigate to your new VPC, choose **Edit DHCP options set** under the **Actions** menu, and select the new option set. You cannot edit the domain name using the console after the DNS option set has been created.

    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*.
16. After the VPC is created, go to the **Subnets** page and note the **Subnet ID** of one of the subnets of your new VPC. You use this information when you launch the Amazon 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. 532).

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/emr.
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 Amazon 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 Amazon 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 Amazon 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 m5.xlarge --instance-count 3
```

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

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

For more information about using Amazon EMR commands in the AWS CLI, see the AWS CLI.

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. This private subnet policy is a part of the VPC endpoint policies for accessing Amazon S3. With Amazon EMR 5.25.0 or later, to enable one-click access to persistent Spark history server, you must allow Amazon EMR to access the system bucket that collects Spark event logs. If you enable logging, provide PUT permissions to a `aws157-logs-*` bucket. For more information, see One-click access to persistent Spark History Server.

It is up to you to determine the policy restrictions that meet your business needs. For example, you can specify the Region `packages.us-east-1.amazonaws.com` to avoid an ambiguous Amazon S3 bucket name. The following example policy provides permissions to access Amazon Linux repositories and the Amazon EMR system bucket for collecting Spark event logs. Replace *MyRegion* with the Region where your log buckets reside, for example `us-east-1`.

For more information about using IAM policies with Amazon VPC endpoints, see Endpoint policies for Amazon S3.

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

The following example policy provides the permissions required to access Amazon Linux 2 repositories. Amazon Linux 2 AMI is the default.

```
{
   "Statement": [
       {
           "Sid": "AmazonLinux2AMIRepositoryAccess",
           "Effect": "Allow",
           "Principal": "*",
           "Action": "s3:GetObject",
           "Resource": [
            "arn:aws:s3:::amazonlinux.MyRegion.amazonaws.com/*",
            "arn:aws:s3:::amazonlinux-2-repos-MyRegion/*"
           ]
       }
   ]
}
```

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 Amazon EMR console, the AWS CLI, or the Amazon 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 Amazon 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 Amazon 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 Amazon EC2 instances. Each node type has a single instance fleet, and using a task instance fleet is optional. You can specify up to five EC2 instance types per fleet, or 30 EC2 instance types per fleet when you create a cluster using the AWS CLI or Amazon EMR API and an allocation strategy (p. 319) for 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 specified 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 instances, and you specify whether it's provisioned as an On-Demand or Spot Instance. Instance fleets also provide additional options for Spot Instance and On-Demand purchases. Spot Instance options include a timeout that specifies an action to take if Spot capacity can't be provisioned, and a preferred allocation strategy (capacity-optimized) for launching Spot Instance fleets. On-Demand Instance fleets can also be launched

using the allocation strategy (lowest-price) option. If you use a service role that is not the EMR default service role, or use an EMR managed policy in your service role, you need to add additional permissions to the custom cluster service role to enable the allocation strategy option. For more information, see Service role for Amazon EMR (EMR role) (p. 398).

For more information about configuring instance fleets, see Configure instance fleets (p. 316).

Uniform instance groups

Uniform instance groups offer a simpler setup than instance fleets. Each Amazon EMR cluster can include up to 50 instance groups: one master instance group that contains one Amazon 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 Amazon EC2 instances. You can scale each instance group by adding and removing Amazon EC2 instances manually, or you can set up automatic scaling. For information about adding and removing instances, see Scaling cluster resources (p. 610).

For more information about configuring uniform instance groups, see Configure uniform instance groups (p. 335).

Topics

- Configure instance fleets (p. 316)
- Use capacity reservations with instance fleets (p. 327)
- Configure uniform instance groups (p. 335)
- Cluster configuration guidelines and best practices (p. 338)

Configure instance fleets

> **Note**
> The instance fleets configuration is available only in Amazon EMR releases 4.8.0 and later, excluding 5.0.0 and 5.0.3.

The instance fleet configuration for Amazon EMR clusters lets you select a wide variety of provisioning options for Amazon EC2 instances, and helps you develop a flexible and elastic resourcing strategy for each node type in your cluster.

In an instance fleet configuration, you specify a *target capacity* for On-Demand Instances and Spot Instances within each fleet. When the cluster launches, Amazon EMR provisions instances until the targets are fulfilled. When Amazon EC2 reclaims a Spot Instance in a running cluster because of a price increase or instance failure, 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.

You can specify a maximum of five Amazon EC2 instance types per fleet for Amazon EMR to use when fulfilling the targets, or a maximum of 30 Amazon EC2 instance types per fleet when you create a cluster using the AWS CLI or Amazon EMR API and an allocation strategy (p. 319) for On-Demand and Spot Instances.

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. If Amazon EMR detects an AWS large-scale event in one or more of the Availability Zones, Amazon EMR automatically attempts to route traffic away from the impacted Availability Zones and tries to launch new clusters that you create in alternate Availability Zones according to your selections. Note that cluster Availability Zone selection happens only at cluster creation. Existing cluster nodes are not automatically re-launched in a new Availability Zone in the event of an Availability Zone outage.

Considerations

Consider the following items when you use instance fleets with Amazon EMR.

- You can have one instance fleet, and only one, per node type (master, core, task). You can specify up to five Amazon EC2 instance types for each fleet on the AWS Management Console (or a maximum of 30 types per instance fleet when you create a cluster using the AWS CLI or Amazon EMR API and an Allocation strategy for instance fleets (p. 319)).
- Amazon EMR chooses any or all of the specified Amazon EC2 instance types to provision with both Spot and On-Demand purchasing options.
- You can 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 Amazon 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.
- You can choose one subnet (Availability Zone) or a range. If you choose 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.
  - 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. This only applies for provisioning clusters, not resizing them. If the timeout period ends during the cluster resizing process, unprovisioned Spot requests will be nullified without transferring to On-Demand capacity.
- For each fleet, you can choose to apply an allocation strategy - lowest-price for On-Demand Instances; capacity-optimized for Spot Instances.
- For each fleet with On-Demand `allocation strategy - lowest-price`, you can choose to apply capacity reservation options.
- Check your subnet size before launching your cluster. When you provision a cluster with a task fleet and there aren't enough IP addresses available in the corresponding subnet, the fleet will go into a suspended state instead of terminating the cluster with an error. To avoid this issue, we recommend increasing the number of IP addresses in your subnets.

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 Amazon 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 Amazon EC2 instance types. If you use an Allocation strategy for instance fleets (p. 319) and create a cluster using the AWS CLI or the Amazon EMR API, you can

specify up to 30 EC2 instance types per instance fleet. 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. 640). 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.

Launch options

For Spot Instances, you can specify a **Maximum Spot price** for each instance type 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.

> **Note**
> Spot Instances with a defined duration (also known as Spot blocks) are no longer available to new customers from July 1, 2021. For customers who have previously used the feature, we will continue to support Spot Instances with a defined duration until December 31, 2022.

For each fleet, you also define a **Provisioning timeout**. The timeout applies when a new cluster is provisioning capacity 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. This only applies for provisioning clusters, not resizing them. If the timeout period ends during the cluster resizing process, unprovisioned Spot requests will be nullified without transferring to On-Demand capacity.

Available in Amazon EMR 5.12.1 and later, you have the option to launch Spot and On-Demand Instance fleets with optimized capacity allocation. This allocation strategy option can be set in the AWS Management Console or using the API `RunJobFlow`. Using the allocation strategy option requires additional service role permissions. If you use the default Amazon EMR service role and managed policy (EMR_DefaultRole (p. 398) and AmazonEMRServicePolicy_v2) for the cluster, the permissions for the allocation strategy option are already included. If you're not using the default Amazon EMR service role and managed policy, you must add them to use this option. See Service role for Amazon EMR (EMR role) (p. 398).

For more information about Spot Instances, see Spot Instances in the Amazon EC2 User Guide for Linux Instances. For more information about On-Demand Instances, see On-Demand Instances in the Amazon EC2 User Guide for Linux Instances.

If you choose to launch On-Demand Instance fleets with the lowest-price allocation strategy, you have the option to use capacity reservations. Capacity reservation options can be set using the Amazon EMR API `RunJobFlow`. Capacity reservations require additional service role permissions which you must add to use these options. See Required IAM permissions to use an allocation strategy (p. 319).

Multiple subnet (Availability Zones) options

When you use instance fleets, you can specify multiple Amazon 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 (or a maximum of 30 when you use the allocation strategy option for On-Demand or Spot Instances). 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.

Allocation strategy for instance fleets

With Amazon EMR versions 5.12.1 and later, you can use the allocation strategy option with On-Demand and Spot Instances for each cluster node. When you create a cluster using the AWS CLI or Amazon EMR API and an allocation strategy option, you can specify a maximum of 30 Amazon EC2 instance types per fleet, as opposed to the five allowed when you use the default Amazon EMR cluster instance fleet configuration. We recommend that you use the allocation strategy option for faster cluster provisioning, more accurate Spot Instance allocation, and fewer Spot Instance interruptions.

- **On-Demand Instances** use a lowest-price strategy, which launches the lowest-priced instances first. When you launch On-Demand Instances, you have the option to use open or targeted capacity reservations in your accounts. You can use open capacity reservations for master, core and task nodes. You may experience insufficient capacity when using On-Demand Instances with allocation strategy for instance fleets. We recommend specifying a larger number of instance types to diversify and reduce the chance of experiencing insufficient capacity. For more information, see Use capacity reservations with instance fleets (p. 327).
- **Spot Instances** use a capacity-optimized strategy, which launches Spot Instances from Spot Instance pools that have optimal capacity for the number of instances that are launching.

Using the allocation strategy option requires several IAM permissions that are automatically included in the default Amazon EMR service role and Amazon EMR managed policy (EMR_DefaultRole and AmazonEMRServicePolicy_v2). If you use a custom service role or managed policy for your cluster, you must add these permissions before you create the cluster. For more information, see Required IAM permissions to use an allocation strategy (p. 319).

Optional On-Demand Capacity Reservations (ODCRs) are available when you use the On-Demand allocation strategy option. Capacity reservation options let you specify a preference for using reserved capacity first for Amazon EMR clusters. You can use this to ensure that your critical workloads use the capacity you have already reserved using open or targeted ODCRs. For non-critical workloads, the capacity reservation preferences let you specify whether reserved capacity should be consumed.

Capacity reservations can only be used by instances that match their attributes (instance type, platform, and Availability Zone). By default, open capacity reservations are automatically used by Amazon EMR when provisioning On-Demand Instances that match the instance attributes. If you don't have any running instances that match the attributes of the capacity reservations, they remain unused until you launch an instance matching their attributes. If you don't want to use any capacity reservations when launching your cluster, you must set capacity reservation preference to **none** in launch options.

However, you can also target a capacity reservation for specific workflows. This enables you to explicitly control which instances are allowed to run in that reserved capacity. For more information about On-Demand Capacity Reservations, see Use capacity reservations with instance fleets (p. 327).

Required IAM permissions to use an allocation strategy

Your Service role for Amazon EMR (EMR role) (p. 398) requires additional permissions to create a cluster that uses the allocation strategy option for On-Demand or Spot Instance fleets.

The required permissions are automatically included in the default Amazon EMR service role and Amazon EMR managed policy (EMR_DefaultRole (p. 398) and AmazonEMRServicePolicy_v2). If you use a custom service role or managed policy for your cluster, you must add the following permissions.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "ec2:DeleteLaunchTemplate",
    "ec2:CreateLaunchTemplate",
    "ec2:DescribeLaunchTemplates",
    "ec2:CreateLaunchTemplateVersion",
    "ec2:CreateFleet"
   ],
   "Resource": "*"
  }
}
```

Following are the service role permissions required to create a cluster that uses open or targeted capacity reservations. You must include these permissions in addition to the permissions required for using the allocation strategy option.

Example Policy document for service role capacity reservations

To use open capacity reservations, you must include the following additional permissions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeCapacityReservations",
                "ec2:DescribeLaunchTemplateVersions",
                "ec2:DeleteLaunchTemplateVersions"
            ],
            "Resource": "*"
        }
    ]
}
```

Example

To use targeted capacity reservations, you must include the following additional permissions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeCapacityReservations",
                "ec2:DescribeLaunchTemplateVersions",
                "ec2:DeleteLaunchTemplateVersions",
                "resource-groups:ListGroupResources"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

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/emr.
2. Choose **Create cluster**.
3. At the top of the console window, choose **Go to advanced options**, enter **Software Configuration** options, and then choose **Next**.
4. Under **Cluster Composition**, choose **Instance fleets**. When you select the instance fleets option, you should see options to specify the **Target capacity** of On-demand and Spot Instances appear in the **Cluster Nodes and Instances** table.
5. For **Network**, enter a value. If you choose a VPC for **Network**, choose a single **EC2 Subnet** or CTRL + click to choose multiple Amazon 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. Under **Allocation Strategy**, select the check box to apply allocation strategies.

   If you don't want to use the new allocation strategy option, leave the check box unchecked.
7. For each **Node type**, if you want to change the default name of an instance fleet, choose the pencil icon and then enter a friendly name. If want to remove the **Task** instance fleet, choose the X icon on the right side of the Task row.
8. Choose **Add/remove instance types to fleet** and choose up to five instance types from the list for master and core instance fleets; add up to fifteen instance types for task instance fleets. Amazon EMR may choose to provision any mix of these instance types when it launches the cluster.
9. For each core and task instance type, choose how you want to define the weighted capacity (**Each instance counts as X units**) for that instance. The number of YARN vCores for each Fleet instance type is used as the default weighted capacity units, but you can change the value to any units that make sense for your applications.
10. Under **Target capacity**, define the total number of On-Demand and Spot Instances you want per fleet. EMR ensures that instances in the fleet fulfill the requested units for On-Demand and Spot target capacity. If no On-Demand or Spot units are specified for a fleet, then no capacity is provisioned for that fleet.
11. If a fleet is configured with a Target capacity for Spot, you can enter your maximum Spot price as a % of On-Demand pricing, or you can enter a Dollars ($) amount in USD.
12. To have EBS volumes attached to the instance type when it's provisioned, choose the pencil next to EBS Storage and then enter EBS configuration options.
13. If you established an instant count for **Spot units**, set **Advanced Spot options** according to the following guidelines:

    - **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 assigned 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.

14. Choose **Next**, modify other cluster settings, and then choose **Next**.
15. If you selected to apply the new allocation strategy option, in the **Security Options** settings, select an **EMR role** and **EC2 instance profile** that contain the permissions required for the allocation strategy option. Otherwise, the cluster creation will fail.
16. Choose **Create 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 about the instance fleets in a cluster, use the `list-instance-fleets` command.
- To add multiple custom Amazon Linux AMIs to a cluster you're creating, use the `CustomAmiId` option with each `InstanceType` specification. You can configure instance fleet nodes with multiple instance types and multiple custom AMIs to fit your requirements. See Create a cluster with the instance fleets configuration (p. 322).
- 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.
- Multiple custom AMIs can be added to the task instance fleet using the CustomAmiId argument with the add-instance-fleet command. See Create a cluster with the instance fleets configuration (p. 322).
- To use the allocation strategy option when creating an instance fleet, update the service role to include the example policy document in the following section.
- To use the capacity reservations options when creating an instance fleet with On-Demand allocation strategy, update the service role to include the example policy document in the following section.
- The instance fleets are automatically included in the default EMR service role and Amazon EMR managed policy (EMR_DefaultRole and AmazonEMRServicePolicy_v2). If you are using a custom service role or custom managed policy for your cluster, you must add the new permissions for allocation strategy in the following section.

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 Amazon 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=m5.xlarge}']
 \

 InstanceFleetType=CORE,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=m5.xlarge}']
```

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=m5.xlarge,BidPrice=0.5}'] \
    InstanceFleetType=CORE,TargetSpotCapacity=1,\
InstanceTypeConfigs=['{InstanceType=m5.xlarge,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=m5.xlarge}'] \
    InstanceFleetType=CORE,TargetOnDemandCapacity=2,TargetSpotCapacity=6,\
InstanceTypeConfigs=['{InstanceType=m5.xlarge,BidPrice=0.5,WeightedCapacity=2}']
```

Example Example: On-Demand master, spot core with multiple weighted instance Types, 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=m5.xlarge}'] \
    InstanceFleetType=CORE,TargetSpotCapacity=11,\
InstanceTypeConfigs=['{InstanceType=m5.xlarge,BidPrice=0.5,WeightedCapacity=3}',\
'{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, 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=m5.xlarge}']
 \
    InstanceFleetType=CORE,TargetOnDemandCapacity=8,TargetSpotCapacity=6,\
InstanceTypeConfigs=['{InstanceType=m5.xlarge,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=m5.xlarge,BidPrice=0.5,WeightedCapacity=3}']
```

Example Example: Spot master, no core or task, Amazon EBS configuration, default VPC

```
aws emr create-cluster --release-label Amazon 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=m5.xlarge,BidPrice=0.5,\
EbsConfiguration={EbsOptimized=true,EbsBlockDeviceConfigs=[{VolumeSpecification={VolumeType=gp2,
\
SizeIn GB=100}},{VolumeSpecification={VolumeType=io1,SizeInGB=100,Iop
 s=100},VolumesPerInstance=4}]}}']
```

Example Multiple custom AMIs, multiple instance types, on-demand master, on-demand core

```
aws emr create-cluster --release-label Amazon EMR 5.3.1 --service-role EMR_DefaultRole \
  --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole \
  --instance-fleets \
    InstanceFleetType=MASTER,TargetOnDemandCapacity=1,\
InstanceTypeConfigs=['{InstanceType=m5.xlarge,CustomAmiId=ami-123456},
{InstanceType=m6g.xlarge, CustomAmiId=ami-234567}'] \
    InstanceFleetType=CORE,TargetOnDemandCapacity=1,\
InstanceTypeConfigs=['{InstanceType=m5.xlarge,CustomAmiId=ami-123456},
{InstanceType=m6g.xlarge, CustomAmiId=ami-234567}']
```

Example Add a task node to a running cluster with multiple instance types and multiple custom AMIs

```
aws emr add-instance-fleet --cluster-id j-123456 --release-label Amazon EMR 5.3.1 \
  --service-role EMR_DefaultRole \
  --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole \
  --instance-fleet \
    InstanceFleetType=Task,TargetSpotCapacity=1,\
InstanceTypeConfigs=['{InstanceType=m5.xlarge,CustomAmiId=ami-123456}',\
'{InstanceType=m6g.xlarge,CustomAmiId=ami-234567}']
```

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-5.30.0 --service-role EMR_DefaultRole \
--ec2-attributes InstanceProfile=EMR_EC2_DefaultRole \
--instance-fleets file:////my-fleet-config.json
```

The *my-fleet-config.json* file 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": "m5.xlarge",
```

```
                "BidPrice": "0.89"
            }
        ]
    },
    {
        "Name": "Corefleet",
        "InstanceFleetType": "CORE",
        "TargetSpotCapacity": 1,
        "TargetOnDemandCapacity": 1,
        "LaunchSpecifications": {
          "OnDemandSpecification": {
            "AllocationStrategy": "lowest-price",
            "CapacityReservationOptions":
            {
                "UsageStrategy": "use-capacity-reservations-first",
                "CapacityReservationResourceGroupArn": "String"
            }
        },
            "SpotSpecification": {
                "AllocationStrategy": "capacity-optimized",
                "TimeoutDurationMinutes": 120,
                "TimeoutAction": "TERMINATE_CLUSTER"
            }
        },
        "InstanceTypeConfigs": [
            {
                "InstanceType": "m5.xlarge",
                "BidPriceAsPercentageOfOnDemandPrice": 100
            }
        ]
    },
    {
        "Name": "Taskfleet",
        "InstanceFleetType": "TASK",
        "TargetSpotCapacity": 1,
        "LaunchSpecifications": {
          "OnDemandSpecification": {
            "AllocationStrategy": "lowest-price",
            "CapacityReservationOptions":
            {
                "CapacityReservationPreference": "none"
            }
        },
            "SpotSpecification": {
                "TimeoutDurationMinutes": 120,
                "TimeoutAction": "TERMINATE_CLUSTER"
            }
        },
        "InstanceTypeConfigs": [
            {
                "InstanceType": "m5.xlarge",
                "BidPrice": "0.89"
            }
        ]
    }
]
```

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 <cluster-id> \
```

```
  --instance-fleet \
    InstanceFleetId='<instance-fleet-id>',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 <cluster-id>
  --instance-fleet \
    InstanceFleetType=TASK,TargetSpotCapacity=1,\
LaunchSpecifications={SpotSpecification='{TimeoutDurationMinutes=20,TimeoutAction=TERMINATE_CLUSTER}'},
\
InstanceTypeConfigs=['{InstanceType=m5.xlarge,BidPrice=0.5}']
```

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 <cluster-id>
```

```
{
    "InstanceFleets": [
        {
            "Status": {
                "Timeline": {
                    "ReadyDateTime": 1488759094.637,
                    "CreationDateTime": 1488758719.817
                },
                "State": "RUNNING",
                "StateChangeReason": {
                    "Message": ""
                }
            },
            "ProvisionedSpotCapacity": 6,
            "Name": "CORE",
            "InstanceFleetType": "CORE",
            "LaunchSpecifications": {
                "SpotSpecification": {
                    "TimeoutDurationMinutes": 60,
                    "TimeoutAction": "TERMINATE_CLUSTER"
                }
            },
            "ProvisionedOnDemandCapacity": 2,
            "InstanceTypeSpecifications": [
                {
                    "BidPrice": "0.5",
                    "InstanceType": "m5.xlarge",
                    "WeightedCapacity": 2
                }
            ],
            "Id": "if-1ABC2DEFGHIJ3"
        },
        {
            "Status": {
                "Timeline": {
                    "ReadyDateTime": 1488759058.598,
                    "CreationDateTime": 1488758719.811
```

```
                },
                "State": "RUNNING",
                "StateChangeReason": {
                    "Message": ""
                }
            },
            "ProvisionedSpotCapacity": 0,
            "Name": "MASTER",
            "InstanceFleetType": "MASTER",
            "ProvisionedOnDemandCapacity": 1,
            "InstanceTypeSpecifications": [
                {
                    "BidPriceAsPercentageOfOnDemandPrice": 100.0,
                    "InstanceType": "m5.xlarge",
                    "WeightedCapacity": 1
                }
            ],
           "Id": "if-2ABC4DEFGHIJ4"
        }
    ]
}
```

Use capacity reservations with instance fleets

To launch On-Demand Instance fleets with capacity reservations options, attach additional service role permissions which are required to use capacity reservation options. Since capacity reservation options must be used together with On-Demand allocation strategy, you also have to include the permissions required for allocation strategy in your service role and managed policy. For more information, see Required IAM permissions to use an allocation strategy (p. 319).

Amazon EMR supports both open and targeted capacity reservations. The following topics show instance fleets configurations that you can use with the `RunJobFlow` action or `create-cluster` command to launch instance fleets using On-Demand Capacity Reservations.

Use open capacity reservations on a best-effort basis

If the cluster's On-Demand Instances match the attributes of open capacity reservations (instance type, platform, tenancy and Availability Zone) available in your account, the capacity reservations are applied automatically. However, it is not guaranteed that your capacity reservations will be used. For provisioning the cluster, Amazon EMR evaluates all the instance pools specified in the launch request and uses the one with the lowest price that has sufficient capacity to launch all the requested core nodes. Available open capacity reservations that match the instance pool are applied automatically. If available open capacity reservations do not match the instance pool, they remain unused.

Once the core nodes are provisioned, the Availability Zone is selected and fixed. Amazon EMR provisions task nodes into instance pools, starting with the lowest-priced ones first, in the selected Availability Zone until all the task nodes are provisioned. Available open capacity reservations that match the instance pools are applied automatically.

The following are use cases of Amazon EMR capacity allocation logic for using open capacity reservations on a best-effort basis.

Example 1: Lowest-price instance pool in launch request has available open capacity reservations

In this case, Amazon EMR launches capacity in the lowest-price instance pool with On-Demand Instances. Your available open capacity reservations in that instance pool are used automatically.

On-Demand Strategy	lowest-price

Requested Capacity	100		
Instance Type	c5.xlarge	m5.xlarge	r5.xlarge
Available Open capacity reservations	150	100	100
On-Demand Price	$	$$	$$$
Instances Provisioned	100	-	-
Open capacity reservation used	100	-	-
Available Open capacity reservations	50	100	100

After the instance fleet is launched, you can run `describe-capacity-reservations` to see how many unused capacity reservations remain.

Example 2: Lowest-price instance pool in launch request does not have available open capacity reservations

In this case, Amazon EMR launches capacity in the lowest-price instance pool with On-Demand Instances. However, your open capacity reservations remain unused.

On-Demand Strategy	lowest-price		
Requested Capacity	100		
Instance Type	c5.xlarge	m5.xlarge	r5.xlarge
Available Open capacity reservations	-	-	100
On-Demand Price	$	$$	$$$
Instances Provisioned	100	-	-
Open capacity reservation used	-	-	-
Available Open capacity reservations	-	-	100

Configure Instance Fleets to use open capacity reservations on best-effort basis

When you use the `RunJobFlow` action to create an instance fleet-based cluster, set the On-Demand allocation strategy to `lowest-price` and `CapacityReservationPreference` for capacity reservations options to open. Alternately, if you leave this field blank, Amazon EMR defaults the On-Demand Instance's capacity reservation preference to open.

```
"LaunchSpecifications":
    {"OnDemandSpecification": {
        "AllocationStrategy": "lowest-price",
        "CapacityReservationOptions":
         {
            "CapacityReservationPreference": "open"
```

```
        }
       }
    }
```

You can also use the Amazon EMR CLI to create an instance fleet-based cluster using open capacity reservations.

```
aws emr create-cluster \
 --name 'open-ODCR-cluster' \
 --release-label emr-5.30.0 \
 --service-role EMR_DefaultRole \
 --ec2-attributes SubnetId=subnet-22XXXX01,InstanceProfile=EMR_EC2_DefaultRole \
 --instance-fleets
 InstanceFleetType=MASTER,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=c4.xlarge}']
 \

 InstanceFleetType=CORE,TargetOnDemandCapacity=100,InstanceTypeConfigs=['{InstanceType=c5.xlarge},
{InstanceType=m5.xlarge},{InstanceType=r5.xlarge}'],\
   LaunchSpecifications={OnDemandSpecification='{AllocationStrategy=lowest-
price,CapacityReservationOptions={CapacityReservationPreference=open}}'}
```

Where,

- `open-ODCR-cluster` is replaced with the name of the cluster using open capacity reservations.
- `subnet-22XXXX01` is replaced with the subnet ID.

Use open capacity reservations first

You can choose to override the lowest-price allocation strategy and prioritize using available open capacity reservations first while provisioning an Amazon EMR cluster. In this case, Amazon EMR evaluates all the instance pools with capacity reservations specified in the launch request and uses the one with the lowest price that has sufficient capacity to launch all the requested core nodes. If none of the instance pools with capacity reservations have sufficient capacity for the requested core nodes, Amazon EMR falls back to the best-effort case described in the previous topic. That is, Amazon EMR re-evaluates all the instance pools specified in the launch request and uses the one with the lowest price that has sufficient capacity to launch all the requested core nodes. Available open capacity reservations that match the instance pool are applied automatically. If available open capacity reservations do not match the instance pool, they remain unused.

Once the core nodes are provisioned, the Availability Zone is selected and fixed. Amazon EMR provisions task nodes into instance pools with capacity reservations, starting with the lowest-priced ones first, in the selected Availability Zone until all the task nodes are provisioned. Amazon EMR uses the available open capacity reservations available across each instance pool in the selected Availability Zone first, and only if required, uses the lowest-price strategy to provision any remaining task nodes.

The following are use cases of Amazon EMR capacity allocation logic for using open capacity reservations first.

Example 1: Instance pool with available open capacity reservations in launch request has sufficient capacity for core nodes

In this case, Amazon EMR launches capacity in the instance pool with available open capacity reservations regardless of instance pool price. As a result, your open capacity reservations are used whenever possible, until all core nodes are provisioned.

On-Demand Strategy	lowest-price

Requested Capacity	100		
Usage Strategy	use-capacity-reservations-first		
Instance Type	c5.xlarge	m5.xlarge	r5.xlarge
Available Open capacity reservations	-	-	150
On-Demand Price	$	$$	$$$
Instances Provisioned	-	-	100
Open capacity reservation used	-	-	100
Available Open capacity reservations	-	-	50

Example 2: Instance pool with available open capacity reservations in launch request does not have sufficient capacity for core nodes

In this case, Amazon EMR falls back to launching core nodes using lowest-price strategy with a best-effort to use capacity reservations.

On-Demand Strategy	lowest-price		
Requested Capacity	100		
Usage Strategy	use-capacity-reservations-first		
Instance Type	c5.xlarge	m5.xlarge	r5.xlarge
Available Open capacity reservations	10	50	50
On-Demand Price	$	$$	$$$
Instances Provisioned	100	-	-
Open capacity reservation used	10	-	-
Available open capacity reservations	-	50	50

After the instance fleet is launched, you can run `describe-capacity-reservations` to see how many unused capacity reservations remain.

Configure Instance Fleets to use open capacity reservations first

When you use the `RunJobFlow` action to create an instance fleet-based cluster, set the On-Demand allocation strategy to `lowest-price` and `UsageStrategy` for `CapacityReservationOptions` to `use-capacity-reservations-first`.

```
"LaunchSpecifications":
    {"OnDemandSpecification": {
        "AllocationStrategy": "lowest-price",
        "CapacityReservationOptions":
```

```
        {
           "UsageStrategy": "use-capacity-reservations-first"
        }
      }
    }
```

You can also use the Amazon EMR CLI to create an instance-fleet based cluster using capacity reservations first.

```
aws emr create-cluster \
  --name 'use-CR-first-cluster' \
  --release-label emr-5.30.0 \
  --service-role EMR_DefaultRole \
  --ec2-attributes SubnetId=subnet-22XXXX01,InstanceProfile=EMR_EC2_DefaultRole \
  --instance-fleets \

 InstanceFleetType=MASTER,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=c4.xlarge}']
 \

 InstanceFleetType=CORE,TargetOnDemandCapacity=100,InstanceTypeConfigs=['{InstanceType=c5.xlarge},
{InstanceType=m5.xlarge},{InstanceType=r5.xlarge}'],\
LaunchSpecifications={OnDemandSpecification='{AllocationStrategy=lowest-
price,CapacityReservationOptions={UsageStrategy=use-capacity-reservations-first}}'}
```

Where,

- `use-CR-first-cluster` is replaced with the name of the cluster using open capacity reservations.
- `subnet-22XXXX01` is replaced with the subnet ID.

Use targeted capacity reservations first

When you provision an Amazon EMR cluster, you can choose to override the lowest-price allocation strategy and prioritize using available targeted capacity reservations first. In this case, Amazon EMR evaluates all the instance pools with targeted capacity reservations specified in the launch request and picks the one with the lowest price that has sufficient capacity to launch all the requested core nodes. If none of the instance pools with targeted capacity reservations have sufficient capacity for core nodes, Amazon EMR falls back to the best-effort case described earlier. That is, Amazon EMR re-evaluates all the instance pools specified in the launch request and selects the one with the lowest price that has sufficient capacity to launch all the requested core nodes. Available open capacity reservations which match the instance pool get applied automatically. However, targeted capacity reservations remain unused.

Once the core nodes are provisioned, the Availability Zone is selected and fixed. Amazon EMR provisions task nodes into instance pools with targeted capacity reservations, starting with the lowest-priced ones first, in the selected Availability Zone until all the task nodes are provisioned. Amazon EMR tries to use the available targeted capacity reservations available across each instance pool in the selected Availability Zone first. Then, only if required, Amazon EMR uses the lowest-price strategy to provision any remaining task nodes.

The following are use cases of Amazon EMR capacity allocation logic for using targeted capacity reservations first.

Example 1: Instance pool with available targeted capacity reservations in launch request has sufficient capacity for core nodes

In this case, Amazon EMR launches capacity in the instance pool with available targeted capacity reservations regardless of instance pool price. As a result, your targeted capacity reservations are used whenever possible until all core nodes are provisioned.

On-Demand Strategy	lowest-price		
Usage Strategy	use-capacity-reservations-first		
Requested Capacity	100		
Instance Type	c5.xlarge	m5.xlarge	r5.xlarge
Available targeted capacity reservations	-	-	150
On-Demand Price	$	$$	$$$
Instances Provisioned	-	-	100
Targeted capacity reservation used	-	-	100
Available targeted capacity reservations	-	-	50

Example Example 2: Instance pool with available targeted capacity reservations in launch request does not have sufficient capacity for core nodes

On-Demand Strategy	lowest-price		
Requested Capacity	100		
Usage Strategy	use-capacity-reservations-first		
Instance Type	c5.xlarge	m5.xlarge	r5.xlarge
Available targeted capacity reservations	10	50	50
On-Demand Price	$	$$	$$$
Instances Provisioned	100	-	-
Targeted capacity reservations Used	-	-	-
Available targeted capacity reservations	10	50	50

After the instance fleet is launched, you can run `describe-capacity-reservations` to see how many unused capacity reservations remain.

Configure Instance Fleets to use targeted capacity reservations first

When you use the `RunJobFlow` action to create an instance-fleet based cluster, set the On-Demand allocation strategy to `lowest-price`, `UsageStrategy` for `CapacityReservationOptions` to `use-capacity-reservations-first`, and `CapacityReservationResourceGroupArn` for `CapacityReservationOptions` to `<your resource group ARN>`. For more information, see Work with capacity reservations in the *Amazon EC2 User Guide for Linux Instances*.

```
"LaunchSpecifications":
    {"OnDemandSpecification": {
```

```
        "AllocationStrategy": "lowest-price",
        "CapacityReservationOptions":
         {
            "UsageStrategy": "use-capacity-reservations-first",
            "CapacityReservationResourceGroupArn": "arn:aws:resource-groups:sa-
east-1:123456789012:group/MyCRGroup"
         }
       }
    }
```

Where `arn:aws:resource-groups:sa-east-1:123456789012:group/MyCRGroup` is replaced with your resource group ARN.

You can also use the Amazon EMR CLI to create an instance fleet-based cluster using targeted capacity reservations.

```
aws emr create-cluster \
  --name 'targeted-CR-cluster' \
  --release-label emr-5.30.0 \
  --service-role EMR_DefaultRole \
  --ec2-attributes SubnetId=subnet-22XXXX01,InstanceProfile=EMR_EC2_DefaultRole \
  --instance-fleets
 InstanceFleetType=MASTER,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=c4.xlarge}']
 \
    InstanceFleetType=CORE,TargetOnDemandCapacity=100,\
InstanceTypeConfigs=['{InstanceType=c5.xlarge},{InstanceType=m5.xlarge},
{InstanceType=r5.xlarge}'],\
LaunchSpecifications={OnDemandSpecification='{AllocationStrategy=lowest-
price,CapacityReservationOptions={UsageStrategy=use-capacity-reservations-
first,CapacityReservationResourceGroupArn=arn:aws:resource-groups:sa-
east-1:123456789012:group/MyCRGroup}}'}
```

Where,

- `targeted-CR-cluster` is replaced with the name of your cluster using targeted capacity reservations.
- `subnet-22XXXX01` is replaced with the subnet ID.
- `arn:aws:resource-groups:sa-east-1:123456789012:group/MyCRGroup` is replaced with your resource group ARN.

Avoid using available open capacity reservations

Example

If you want to avoid unexpectedly using any of your open capacity reservations when launching an Amazon EMR cluster, set the On-Demand allocation strategy to `lowest-price` and `CapacityReservationPreference` for `CapacityReservationOptions` to none. Otherwise, Amazon EMR defaults the On-Demand Instance's capacity reservation preference to open and tries using available open capacity reservations on a best-effort basis.

```
"LaunchSpecifications":
    {"OnDemandSpecification": {
        "AllocationStrategy": "lowest-price",
        "CapacityReservationOptions":
         {
            "CapacityReservationPreference": "none"
         }
       }
    }
```

You can also use the Amazon EMR CLI to create an instance fleet-based cluster without using any open capacity reservations.

```
aws emr create-cluster \
  --name 'none-CR-cluster' \
  --release-label emr-5.30.0 \
  --service-role EMR_DefaultRole \
  --ec2-attributes SubnetId=subnet-22XXXX01,InstanceProfile=EMR_EC2_DefaultRole \
  --instance-fleets \

 InstanceFleetType=MASTER,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=c4.xlarge}']
 \

 InstanceFleetType=CORE,TargetOnDemandCapacity=100,InstanceTypeConfigs=['{InstanceType=c5.xlarge},
{InstanceType=m5.xlarge},{InstanceType=r5.xlarge}'],\
LaunchSpecifications={OnDemandSpecification='{AllocationStrategy=lowest-
price,CapacityReservationOptions={CapacityReservationPreference=none}}'}
```

Where,

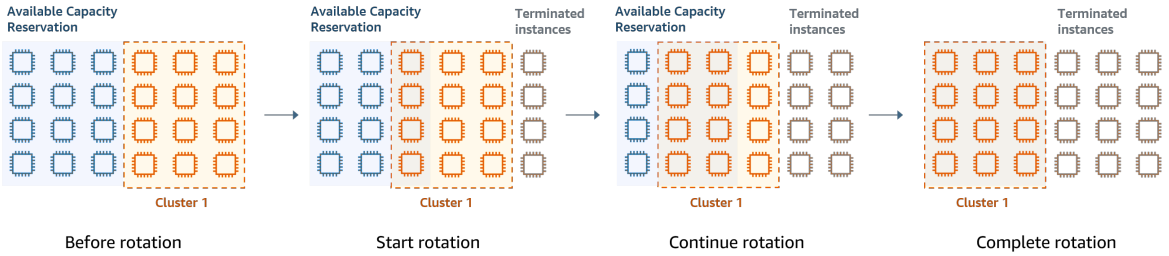
- `none-CR-cluster` is replaced with the name of your cluster that is not using any open capacity reservations.
- `subnet-22XXXX01` is replaced with the subnet ID.

Scenarios for using capacity reservations

You can benefit from using capacity reservations in the following scenarios.

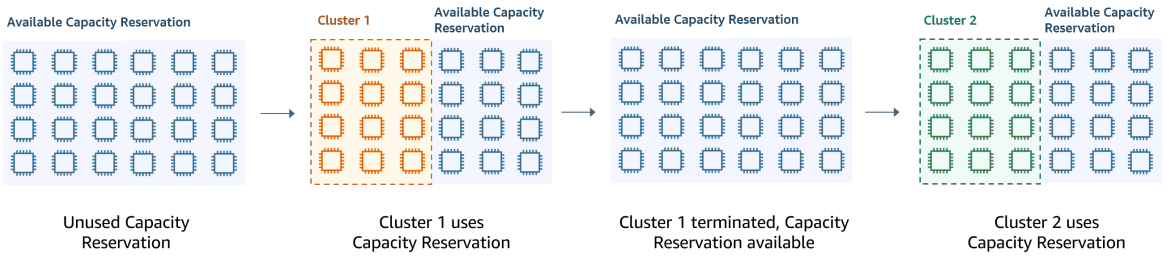
Scenario 1: Rotate a long-running cluster using capacity reservations

When rotating a long running cluster, you might have strict requirements on the instance types and Availability Zones for the new instances you provision. With capacity reservations, you can use capacity assurance to complete the cluster rotation without interruptions.



Scenario 2: Provision successive short-lived clusters using capacity reservations

You can also use capacity reservations to provision a group of successive, short-lived clusters for individual workloads so that when you terminate a cluster, the next cluster can use the capacity reservations. You can use targeted capacity reservations to ensure that only the intended clusters use the capacity reservations.



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.

If the cluster's On-Demand Instances match the attributes of open capacity reservations (instance type, platform, tenancy and Availability Zone) available in your account, the capacity reservations are applied automatically. You can use open capacity reservations for master, core, and task nodes. However, you cannot use targeted capacity reservations or prevent instances from launching into open capacity reservations with matching attributes when you provision clusters using instance groups. If you want to use targeted capacity reservations or prevent instances from launching into open capacity reservations, use Instance Fleets instead. For more information, see Use capacity reservations with instance fleets (p. 327).

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. 610).

When launching instances, the On-Demand Instance's capacity reservation preference defaults to open, which enables it to run in any open capacity reservation that has matching attributes (instance type, platform, Availability Zone). For more information about On-Demand Capacity Reservations, see Use capacity reservations with instance fleets (p. 327).

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. 550).

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/emr.
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. 306).

   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, choose the pencil icon and then enter a friendly name. If want to remove the **Task** instance group, choose the X icon. Choose **Add task instance group** to add additional **Task** instance groups.
   - Under **Instance type**, choose 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 type**, choose the pencil icon for **Configurations** and then edit the configurations for applications for each instance group.
- Under **Instance count**, enter the number of instances to use for each 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
  Pause on 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 with a custom policy for instance groups (p. 625).

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 Instance 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 Amazon 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-5.36.0 \
  --use-default-roles \
  --ec2-attributes KeyName=myKey \
  --instance-groups \
    InstanceGroupType=MASTER,InstanceType=m5.xlarge,InstanceCount=1,BidPrice=0.25 \
    InstanceGroupType=CORE,InstanceType=m5.xlarge,InstanceCount=2,BidPrice=0.03 \
    InstanceGroupType=TASK,InstanceType=m5.xlarge,InstanceCount=4,BidPrice=0.03 \
```

```
    InstanceGroupType=TASK,InstanceType=m5.xlarge,InstanceCount=2,BidPrice=0.04
```

Using the CLI, you can create uniform instance group clusters that specify a unique custom AMI for each instance type in the instance group. This allows you to use different instance architectures in the same instance group. Each instance type must use a custom AMI with a matching architecture. For example, you would configure an m5.xlarge instance type with an x86_64 architecture custom AMI, and an m6g.xlarge instance type with a corresponding `AWS AARCH64` (ARM) architecture custom AMI.

The following example shows a uniform instance group cluster created with two instance types, each with its own custom AMI. Notice that the custom AMIs are specified only at the instance type level, not at the cluster level. This is to avoid conflicts between the instance type AMIs and an AMI at the cluster level, which would cause the cluster launch to fail.

```
aws emr create-cluster
  --release-label emr-5.30.0 \
  --service-role EMR_DefaultRole \
  --ec2-attributes SubnetId=subnet-22XXXX01,InstanceProfile=EMR_EC2_DefaultRole \
  --instance-groups \
    InstanceGroupType=MASTER,InstanceType=m5.xlarge,InstanceCount=1,CustomAmiId=ami-123456
 \
    InstanceGroupType=CORE,InstanceType=m6g.xlarge,InstanceCount=1,CustomAmiId=ami-234567
```

You can add multiple custom AMIs to an instance group that you add to a running cluster. The `CustomAmiId` argument can be used with the `add-instance-groups` command as shown in the following example.

```
aws emr add-instance-groups --cluster-id j-123456 \
  --instance-groups \
    InstanceGroupType=Task,InstanceType=m5.xlarge,InstanceCount=1,CustomAmiId=ami-123456
```

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 Amazon 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. 610).
- **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. 316).

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. 550). 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; Amazon EC2 instances for the core node type, which process tasks and store data in HDFS, need both processing power and storage capacity; Amazon EC2 instances for the task node type, which don't store data, need only processing power. For guidelines about available Amazon EC2 instances and their configuration, see Configure Amazon EC2 instances (p. 197).

The following guidelines apply to most Amazon EMR clusters.

- There is a vCPU limit for the total number of on-demand Amazon EC2 instances that you run on an AWS account per AWS Region. For more information about the vCPU limit and how to request a limit increase for your account, see On-Demand Instances in the *Amazon EC2 User Guide for Linux Instances*.
- The master node does not typically have large computational requirements. For clusters with a large number of nodes, or for clusters with applications that are specifically deployed on the master node (JupyterHub, Hue, etc.), a larger master node may be required and can help improve cluster performance. For example, consider using an m5.xlarge instance for small clusters (50 or fewer nodes), and increasing to a larger instance type for larger clusters.
- The computational needs of the core and task nodes depend on the type of processing your application performs. Many jobs can be run on general purpose instance types, which offer balanced performance in terms of CPU, disk space, and input/output. 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.
- 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.

- 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.

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. 339)
- Master node on a Spot Instance (p. 340)
- Core nodes on Spot Instances (p. 340)
- Task nodes on Spot Instances (p. 340)
- Instance configurations for application scenarios (p. 341)

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 do not 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 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`

  Note
  Beginning with Amazon EMR 6.x release series, the YARN node labels feature is disabled by default. The application master processes can run on both core and task nodes by default. You can enable the YARN node labels feature by configuring following properties:

  - `yarn.node-labels.enabled: true`
  - `yarn.node-labels.am.default-node-label-expression: 'CORE'`

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. 315).

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. 341)	On-Demand	On-Demand or instance-fleet mix	Spot or instance-fleet mix
Cost-driven workloads (p. 341)	Spot	Spot	Spot
Data-critical workloads (p. 341)	On-Demand	On-Demand	Spot or instance-fleet mix
Application testing (p. 341)	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 Instances and supplement with one or more task instance groups of Spot Instances. Running the master and core instance groups as On-Demand Instances 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 the following factors:

- The number of Amazon EC2 instances used for core nodes.

- The capacity of the Amazon EC2 instance store for the instance type used. For more information on instance store volumes, see Amazon Amazon EC2 instance store in the *Amazon EC2 User Guide for Linux Instances*.
- The number and size of Amazon 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 Amazon 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 Amazon EBS volumes, has a total of approximately 2,666 GB available for HDFS (10 nodes x 800 GB ÷ 3 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 Amazon EBS volumes or adding instance groups with attached Amazon EBS volumes to an existing cluster
- Adding more core nodes
- Choosing an Amazon 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 primary node. These are written to the `/mnt/var/log/` directory. You can access them by using SSH to connect to the primary node as described in Connect to the master node using SSH (p. 593).

> **Note**
> If you use Amazon EMR release 6.8.0 or earlier, log files are saved to Amazon S3 during cluster termination, so you can't access the log files once the primary node terminates. Amazon EMR

releases 6.9.0 and later archive logs to Amazon S3 during cluster scale-down, so log files generated on the cluster persist even after the node is terminated.

You do not need to enable anything to have log files written on the primary 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 primary 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 primary 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 primary node.

Archive log files to Amazon S3

> **Note**
> You cannot currently use log aggregation to Amazon S3 with the `yarn logs` utility.

Amazon EMR releases 6.9.0 and later archive logs to Amazon S3 during cluster scale-down, so log files generated on the cluster persist even after the node is terminated. This behavior is enabled automatically, so you don't need to do anything to turn it on. For Amazon EMR releases 6.8.0 and earlier, you can configure a cluster to periodically archive the log files stored on the primary 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 for Amazon EMR releases 6.8.0 and earlier, 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/emr.
2. Choose **Create cluster**.
3. Choose **Go to advanced options**.
4. In the **General options** 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. 564).

5. In the **S3 folder** 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.
6. In the **Log encryption** field, select **Encrypt logs stored in S3 with an AWS KMS customer managed key**. Then select an AWS KMS key from the list or enter a key ARN. You may also create a new KMS key.

   This option is only available with Amazon EMR version 5.30.0 and later. To use this option, add permission to KMS for your EC2 instance profile and EMR role. For more information, see To encrypt log files stored in Amazon S3 with an AWS KMS customer managed key (p. 344).
7. Proceed with creating the cluster as described in Plan and configure clusters (p. 131).

To encrypt log files stored in Amazon S3 with an AWS KMS customer managed key

With Amazon EMR version 5.30.0 and later (except Amazon EMR 6.0.0), you can encrypt log files stored in Amazon S3 with an AWS KMS customer managed key. To enable this option in the console, follow the steps in To archive log files to Amazon S3 using the console (p. 343). Your Amazon EC2 instance profile and your Amazon EMR role must meet the following prerequisites:

- The Amazon EC2 instance profile used for your cluster must have permission to use `kms:GenerateDataKey`.
- The Amazon EMR role used for your cluster must have permission to use `kms:DescribeKey`.
- The Amazon EC2 instance profile and Amazon EMR role must be added to the list of key users for the specified AWS KMS customer managed key, as the following steps demonstrate:
  1. 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 KMS key to modify.
  4. On the key details page under **Key Users**, choose **Add**.
  5. In the **Add key users** dialog box, select your Amazon EC2 instance profile and Amazon EMR role.
  6. Choose **Add**.

For more information, see IAM service roles used by Amazon EMR, and Using key policies in the AWS Key Management Service developer guide.

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-5.36.0 --log-uri
 s3://DOC-EXAMPLE-BUCKET/logs --applications Name=Hadoop Name=Hive Name=Pig --use-
```

```
default-roles --ec2-attributes KeyName=myKey --instance-type m5.xlarge --instance-
count 3
```

When you specify the instance count without using the `--instance-groups` parameter, a single primary 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 AWS CLI Command Reference.

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.

- To enable log aggregation create the following configuration file called `myConfig.json` that 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:\/\/DOC-EXAMPLE-BUCKET\/logs"
    }
  }
]
```

  Type the following command and replace *myKey* with the name of your EC2 key pair. You can additionally replace any of the red text with your own configurations.

```
aws emr create-cluster --name "Test cluster" \
--release-label emr-5.36.0 \
--applications Name=Hadoop \
--use-default-roles \
--ec2-attributes KeyName=myKey \
--instance-type m5.xlarge \
--instance-count 3 \
--configurations file://./myConfig.json
```

  When you specify the instance count without using the `--instance-groups` parameter, a single primary 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, run `aws emr create-default-roles` to create them before running the `create-cluster` subcommand.

For more information on using Amazon EMR commands in the AWS CLI, see AWS CLI Command Reference.

Log locations

The following list includes all log types and their locations in Amazon S3. You can use these for troubleshooting Amazon EMR issues.

Step logs

```
s3://DOC-EXAMPLE-LOG-BUCKET/<cluster-id>/steps/<step-id>/
```

Application logs

```
s3://DOC-EXAMPLE-LOG-BUCKET/<cluster-id>/containers/
```

This location includes container `stderr` and `stdout`, `directory.info`, `prelaunch.out`, and `launch_container.sh` logs.

Resource manager logs

```
s3://DOC-EXAMPLE-LOG-BUCKET/<cluster-id>/node/<leader-instance-id>/
applications/hadoop-yarn/
```

Hadoop HDFS

```
s3://DOC-EXAMPLE-LOG-BUCKET/<cluster-id>/node/<all-instance-id>/
applications/hadoop-hdfs/
```

This location includes NameNode, DataNode, and YARN TimelineServer logs.

Node manager logs

```
s3://DOC-EXAMPLE-LOG-BUCKET/<cluster-id>/node/<all-instance-id>/
applications/hadoop-yarn/
```

Instance-state logs

```
s3://DOC-EXAMPLE-LOG-BUCKET/<cluster-id>/node/<all-instance-id>/daemons/
instance-state/
```

Amazon EMR provisioning logs

```
s3://DOC-EXAMPLE-LOG-BUCKET/<cluster-id>/node/<leader-instance-id>/
provision-node/*
```

Hive logs

```
s3://DOC-EXAMPLE-LOG-BUCKET/<cluster-id>/node/<leader-instance-id>/
applications/hive/*
```

- To find Hive logs on your cluster, remove the asterisk (`*`) and append `/var/log/hive/` to the above link.
- To find HiveServer2 logs, remove the asterisk (`*`) and append `var/log/hive/hiveserver2.log` to the above link.
- To find HiveCLI logs, remove the asterisk (`*`) and append `/var/log/hive/user/hadoop/hive.log` to the above link.
- To find Hive Metastore Server logs, remove the asterisk (`*`) and append `/var/log/hive/user/hive/hive.log` to the above link.

If your failure is in the primary or task node of your Tez application, provide logs of the appropriate Hadoop container.

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. 567). 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/emr.
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 http://aws.amazon.com/https://aws.amazon.com/sqs.
7. Proceed with creating the cluster as described in Plan and configure clusters (p. 131).

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-5.36.0 \
--log-uri s3://DOC-EXAMPLE-BUCKET/logs \
--enable-debugging \
--applications Name=Hadoop Name=Hive Name=Pig \
--use-default-roles \
--ec2-attributes KeyName=myKey \
--instance-type m5.xlarge \
--instance-count 3
```

  When you specify the instance count without using the `--instance-groups` parameter, a single primary 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 AWS CLI Command Reference.

Example Enabling debugging using the Java SDK

Enable debugging using the following StepConfig:

```
    StepFactory stepFactory = new StepFactory();
 StepConfig enabledebugging = new StepConfig()
     .withName("Enable debugging")
     .withActionOnFailure("TERMINATE_JOB_FLOW")
     .withHadoopJarStep(stepFactory.newEnableDebuggingStep());
```

In this example, `new StepFactory()` uses `us-east-1` as the default region. If your cluster is launched in a different region, you need to specify the region by using `new StepFactory("region.elasticmapreduce")`, such as `new StepFactory("ap-northeast-2.elasticmapreduce")`.

Debugging option information

Amazon EMR release 4.1.0 through 5.27.0 supports debugging in all regions. Other EMR versions do not support the debugging option.

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 http://aws.amazon.com/https://aws.amazon.com/sqs.

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 service roles for Amazon EMR permissions to AWS services and resources (p. 394).

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 EC2API or command line, see API and CLI overview in the *Amazon EC2 User Guide for Linux Instances*.

Tag resources for billing

You can use tags for organizing your AWS bill to reflect your own cost structure. To do this, sign up to get your AWS account bill with tag key values included. You can then organize your billing information by tag key values, to see the cost of your combined resources. Although Amazon EMR and Amazon EC2 have different billing statements, the tags on each cluster are also placed on each associated instance so you can use tags to link related Amazon EMR and Amazon EC2 costs.

For example, you can tag several resources with a specific application name, and then organize your billing information to see the total cost of that application across several services. For more information, see Cost allocation and tagging in the *AWS Billing 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. Open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Choose **Create cluster**, **Go to advanced options**.
3. On the **Step 3: General Cluster Settings** page, in the **Tags** section, type a **Key** for your tag.

   When you begin typing the **Key**, a new row automatically appears to provide space for the next new tag.
4. Optionally, type a **Value** for the tag.
5. 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 m5.xlarge --instance-count 3
```

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

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

  For more information on using Amazon EMR commands in the AWS CLI, see 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

1. Enter the `add-tags` subcommand with the `--tag` parameter to assign tags to the cluster ID. You can find the cluster ID using the console or the `list-clusters` command. The `add-tags` subcommand currently accepts only one resource ID.

   For example, to add two tags to a running cluster one with a key named *`costCenter`* with a value of *`marketing`* and another named *`other`* with a value of *`accounting`*, enter the following command and replace *`j-KT4XXXXXXXX1NM`* with your cluster ID.

   ```
   aws emr add-tags --resource-id j-KT4XXXXXXXX1NM --tag "costCenter=marketing" --
   tag "other=accounting"
   ```

   Note that when tags are added using the AWS CLI, there's 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 want to see all tags associated with a cluster, you can view them in the console or at the CLI.

To view the tags on a cluster using the console

1. In the Amazon EMR console, select the **Cluster List** page and click a cluster to view tags.
2. On the **Cluster Details** page, in the **Tags** field, some tags are displayed here. Click **View All/Edit** to display all available tags on the cluster.

To view the tags on a cluster using the AWS CLI

To view the tags on a cluster using the AWS CLI, type the `describe-cluster` subcommand with the `--query` parameter.

- To view a cluster's tags, type the following command and replace *`j-KT4XXXXXXXX1NM`* with your cluster ID.

  ```
  aws emr describe-cluster --cluster-id j-KT4XXXXXX1NM --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-KT4XXXXXX1NM --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. 352)

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. To download and install the latest drivers, see http://awssupportdatasvcs.com/bootstrap-actions/Simba/latest/.

To find older versions of drivers, see http://awssupportdatasvcs.com/bootstrap-actions/Simba/.

Security in Amazon EMR

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

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

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

This documentation helps you understand how to apply the shared responsibility model when using Amazon EMR. When you develop solutions on Amazon EMR, use the following technologies to help secure cluster resources and data according to your business requirements. The topics in this chapter show you how to configure Amazon EMR and use other AWS services to meet your security and compliance objectives.

Security configurations

Security configurations in Amazon EMR are templates for different security setups. 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. 355).

Data protection

You can implement data encryption to help protect data at rest in Amazon S3, data at rest in cluster instance storage, and data in transit. For more information, see Encrypt data at rest and in transit (p. 374).

AWS Identity and Access Management with Amazon EMR

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Amazon EMR resources. IAM is an AWS service that you can use with no additional charge.

- **IAM Identity-Based 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 AWS Identity and Access Management for Amazon EMR (p. 382).
- **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 service roles for Amazon EMR permissions to AWS services and resources (p. 394).

- **IAM roles for EMRFS requests to Amazon S3** – When Amazon EMR accesses Amazon S3, you can specify the IAM role to use based on the user, group, or the location of EMRFS data in Amazon S3. This allows you to precisely control whether cluster users can access files from within Amazon EMR. For more information, see Configure IAM roles for EMRFS requests to Amazon S3 (p. 420).

Kerberos

You can set up Kerberos to provide strong authentication through secret-key cryptography. For more information, see Use Kerberos authentication (p. 448).

Lake Formation

You can use Lake Formation permissions together with the AWS Glue Data Catalog to provide fine-grained, column-level access to databases and tables in the AWS Glue Data Catalog. Lake Formation enables federated single sign-on to EMR Notebooks or Apache Zeppelin from an enterprise identity system. For more information, see Integrate Amazon EMR with AWS Lake Formation (p. 473).

Secure Socket Shell (SSH)

SSH helps provide 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. 447) and Connect to the cluster (p. 591).

Amazon EC2 security groups

Security groups act as a virtual firewall for EMR cluster instances, limiting inbound and outbound network traffic. For more information, see Control network traffic with security groups (p. 532).

Updates to the default Amazon Linux AMI for Amazon EMR

> **Important**
> Amazon EMR clusters that are running Amazon Linux or Amazon Linux 2 AMIs (Amazon Linux Machine Images) use default Amazon Linux behavior, and do not automatically download and install important and critical kernel updates that require a reboot. This is the same behavior as other Amazon EC2 instances running the default Amazon Linux AMI. If new Amazon Linux software updates that require a reboot (such as, kernel, NVIDIA, and CUDA updates) become available after an Amazon EMR version is released, Amazon EMR cluster instances running the default AMI do not automatically download and install those updates. To get kernel updates, you can customize your Amazon EMR AMI to use the latest Amazon Linux AMI.

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 Using the default Amazon Linux AMI for Amazon EMR (p. 177). Note that your networking configuration must allow for HTTP and HTTPS egress to Amazon Linux repositories in Amazon S3, otherwise security updates will not succeed.

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. 355)
- Specify a security configuration for a cluster (p. 373)

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 at rest and in transit (p. 374)
- Use Kerberos authentication (p. 448)
- Configure IAM roles for EMRFS requests to Amazon S3 (p. 420)

To create a security configuration using the console

1. Open the Amazon EMR console at https://console.aws.amazon.com/emr.
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 encrypting data at rest with Amazon EMR (p. 378) and Providing certificates for encrypting data in transit with Amazon EMR encryption (p. 381).

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 at rest and in transit (p. 374).

> **Note**
> When you use AWS KMS, charges apply for the storage and use of encryption keys. For more information, see AWS KMS Pricing.

Specifying encryption options using the console

Choose options under **Encryption** according to the following guidelines.

- Choose options under **At rest encryption** to encrypt data stored within the file system.

  You can choose to encrypt data in Amazon S3, local disks, or both.
- 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 EMR cluster. For key requirements, see Using AWS KMS keys for encryption (p. 379).
  - **CSE-Custom**

    Specifies client-side encryption using a custom client-side root 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**.
  - **AWS KMS key**

    Select this option to specify an AWS KMS key. For **AWS KMS key**, select a key. The key must exist in the same region as your EMR cluster. For more information about key requirements, see Using AWS KMS keys for encryption (p. 379).

    EBS Encryption

    When you specify AWS KMS as your key provider, you can enable EBS encryption to encrypt EBS root device and storage volumes. To enable such option, you must grant the EMR service role `EMR_DefaultRole` with permissions to use the AWS KMS key that you specify. For more

information about key requirements, see Enabling EBS encryption by providing additional permissions for KMS keys (p. 379).

- **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 encrypting data in transit with Amazon EMR encryption (p. 381) 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 encrypting data in transit with Amazon EMR encryption (p. 381) 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 encrypting data in transit with Amazon EMR encryption (p. 381) 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"
   }
  }
 }
}'
```

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"
   }
  }
 }
}'
```

The example below illustrates the following scenario:

- In-transit data encryption is disabled and at-rest data encryption is enabled.
- Amazon S3 encryption is enabled with SSE-KMS.
- Multiple AWS KMS keys are used, one per each S3 bucket, and encryption exceptions are applied to these individual S3 buckets.
- Local disk encryption is disabled.

```
aws emr create-security-configuration --name "MySecConfig" --security-configuration '{
   "EncryptionConfiguration": {
     "AtRestEncryptionConfiguration": {
             "S3EncryptionConfiguration": {
           "EncryptionMode": "SSE-KMS",
           "AwsKmsKey": "arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012",
           "Overrides": [
              {
                "BucketName": "sse-s3-bucket-name",
                "EncryptionMode": "SSE-S3"
              },
              {
                "BucketName": "cse-kms-bucket-name",
                "EncryptionMode": "CSE-KMS",
                "AwsKmsKey": "arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012"
              },
              {
                "BucketName": "sse-kms-bucket-name",
                "EncryptionMode": "SSE-KMS",
                "AwsKmsKey": "arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012"
              }
             ]
             }
           },
     "EnableInTransitEncryption": false,
      "EnableAtRestEncryption": true
```

```
  }
}'
```

The example below illustrates the following scenario:

- In-transit data encryption is disabled and at-rest data encryption is enabled.
- Amazon S3 encryption is enabled with SSE-S3 and local disk encryption is disabled.

```
aws emr create-security-configuration --name "MyS3EncryptionConfig" --security-
configuration '{
    "EncryptionConfiguration": {
        "EnableInTransitEncryption": false,
        "EnableAtRestEncryption": true,
        "AtRestEncryptionConfiguration": {
            "S3EncryptionConfiguration": {
                "EncryptionMode": "SSE-S3"
            }
        }
     }
}'
```

The example below illustrates the following scenario:

- In-transit data encryption is disabled and at-rest data encryption is enabled.
- Local disk encryption is enabled with AWS KMS as the key provider and Amazon S3 encryption is disabled.

```
aws emr create-security-configuration --name "MyLocalDiskEncryptionConfig" --security-
configuration '{
    "EncryptionConfiguration": {
        "EnableInTransitEncryption": false,
        "EnableAtRestEncryption": true,
        "AtRestEncryptionConfiguration": {
            "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 disabled and at-rest data encryption is enabled.
- Local disk encryption is enabled with AWS KMS as the key provider and Amazon S3 encryption is disabled.
- EBS encryption is enabled.

```
aws emr create-security-configuration --name "MyLocalDiskEncryptionConfig" --security-
configuration '{
    "EncryptionConfiguration": {
        "EnableInTransitEncryption": false,
        "EnableAtRestEncryption": true,
        "AtRestEncryptionConfiguration": {
            "LocalDiskEncryptionConfiguration": {
```

```
                "EnableEbsEncryption": true,
                "EncryptionKeyProviderType": "AwsKms",
                "AwsKmsKey": "arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012"
            }
        }
    }
}'
```

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" : "`*`ZipLocation`*`" | "`*`JarLocation`*`"`	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" : "`*`MyClassID`*`"`	Required only if `Custom` is specified for `CertificateProviderType`. *`MyClassID`* specifies a full class name declared in the JAR file, which implements the TLSArtifactsProvider interface. For example, `com.mycompany.MyCertProvider`.
At-rest encryption parameters	

Parameter	Description
`"AtRestEncryptionConfiguration" :`	Specifies a collection of values for at-rest encryption when `EnableAtRestEncryption` 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 key ARN must be specified as the `AwsKmsKey` value. If `CSE-Custom` is specified, `S3Object` and `EncryptionKeyProviderClass` values must be specified.
`"AwsKmsKey" : "`*`MyKeyARN`*`"`	Required only when either `SSE-KMS` or `CSE-KMS` is specified for `EncryptionMode`. *`MyKeyARN`* must be a fully specified ARN to a key (for example, `arn:aws:kms:us-east-1:123456789012:key/12345678-1234-1234-1234-1`
`"S3Object" : "`*`JarLocation`*`"`	Required only when `CSE-Custom` is specified for `CertificateProviderType`. *`JarLocation`* provides the location in Amazon S3 to a JAR file. The format can be a path (for example, `s3://MyConfig/artifacts/MyKeyProvider.jar`) or an ARN (for example, `arn:aws:s3:::Code/MyKeyProvider.jar`).
`"EncryptionKeyProviderClass" : "`*`MyS3KeyClassID`*`"`	Required only when `CSE-Custom` is specified for `EncryptionMode`. *`MyS3KeyClassID`* specifies a full class name of a class declared in the application that implements the EncryptionMaterialsProvider interface; for example, *`com.mycompany.MyS3KeyProvider`*.
Local disk encryption parameters	
`"LocalDiskEncryptionConfiguration"`	Specifies the key provider and corresponding values to be used for local disk encryption.
`"EnableEbsEncryption": true | false`	Specify `true` to enable EBS encryption. EBS encryption encrypts the EBS root device volume and attached storage volumes. To use EBS encryption, you must specify `AwsKms` as your `EncryptionKeyProviderType`.
`"EncryptionKeyProviderType": "AwsKms" | "Custom"`	Specifies the key provider. If `AwsKms` is specified, an KMS key ARN must be specified as the `AwsKmsKey` value. If **`Custom`** is specified, `S3Object` and `EncryptionKeyProviderClass` values must be specified.

Parameter	Description
`"AwsKmsKey : "MyKeyARN"`	Required only when `AwsKms` is specified for `Type`. *`MyKeyARN`* must be a fully specified ARN to a key (for example, `arn:aws:kms:us-east-1:123456789012:key/12345678-1234-1234-1234-4`
`"S3Object" : "JarLocation"`	Required only when `CSE-Custom` is specified for `CertificateProviderType`. *`JarLocation`* provides the location in Amazon S3 to a JAR file. The format can be a path (for example, `s3://MyConfig/artifacts/MyKeyProvider.jar`) or an ARN (for example, `arn:aws:s3:::Code/MyKeyProvider.jar`).
`"EncryptionKeyProviderClass" : "MyLocalDiskKeyClassID"`	Required only when `Custom` is specified for `Type`. *`MyLocalDiskKeyClassID`* specifies a full class name of a class declared in the application that implements the EncryptionMaterialsProvider interface; for example, *`com.mycompany.MyLocalDiskKeyProvider`*.

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 Use Kerberos authentication (p. 448). 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
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.
Provider	**Cluster-dedicated KDC**	Specifies that Amazon EMR creates a KDC on the master node of any cluster that uses this security configuration. You specify the realm name and KDC admin password when you create the cluster. You can reference this KDC from other clusters, if required. Create those clusters using a different security configuration, specify an external KDC, and use the realm name and KDC admin password that you specify for the cluster-dedicated KDC.
	External KDC	Available only with Amazon EMR 5.20.0 and later. Specifies that clusters using this security configuration authenticate Kerberos principals using a KDC server outside the cluster. A KDC is not created on the cluster. When you create the cluster, you specify the realm name and KDC admin password for the external KDC.

Parameter		Description
Ticket Lifetime		Optional. Specifies the period for which a Kerberos ticket issued by the KDC is valid on clusters that use this security configuration. 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 `kinit` from the master node command line to renew after a ticket expires.
Cross-realm trust		Specifies a cross-realm trust between a cluster-dedicated KDC on clusters that use this security configuration and a KDC in a different Kerberos realm. Principals (typically users) from another realm are authenticated to clusters that use this configuration. Additional configuration in the other Kerberos realm is required. For more information, see Tutorial: Configure a cross-realm trust with an Active Directory domain (p. 468).
Cross-realm trust properties	**Realm**	Specifies the Kerberos realm name of the other realm in the trust relationship. By convention, Kerberos realm names are the same as the domain name 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) or IP address 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) or IP address 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 use 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`).
External KDC		Specifies that clusters external KDC is used by the cluster.

Parameter			Description
External KDC properties	**Admin server**		Specifies the fully qualified domain name (FQDN) or IP address of the external admin server. 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 external KDC server. The KDC server and admin server typically run on the same machine with the same FQDN, but use 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`).
	Active Directory Integration		Specifies that Kerberos principal authentication is integrated with a Microsoft Active Directory domain.
	Active Directory integration properties	**Active Directory realm**	Specifies the Kerberos realm name of the Active Directory domain. By convention, Kerberos realm names are typically the same as the domain name but in all capital letters.
		Active Directory domain	Specifies the Active Directory domain name.
		Active Directory server	Specifies the fully qualified domain name (FQDN) of the Microsoft Active Directory domain controller.

Specifying Kerberos settings using the AWS CLI

The following reference table shows JSON parameters for Kerberos settings in a security configuration. For example configurations, see, Configuration examples (p. 461).

Parameter			Description
`"AuthenticationConfiguration": {`			Required for Kerberos. Specifies that an authentication configuration is part of this security configuration.
	`"KerberosConfiguration": {`		Required for Kerberos. Specifies Kerberos configuration properties.

		`"Provider": "ClusterDedicatedKdc",`
		
		—or—
		
		`"Provider: "ExternalKdc",`
			`ClusterDedicatedKdc` specifies that Amazon EMR creates a KDC on the master node of any cluster that uses this security configuration. You specify the realm name and KDC admin password when you create the cluster. You can reference this KDC

Parameter	Description
	from other clusters, if required. Create those clusters using a different security configuration, specify an external KDC, and use the realm name and KDC admin password that you specified when you created the cluster with the cluster-dedicated KDC. *ExternalKdc* specifies that the cluster uses an external KDC. Amazon EMR does not create a KDC on the master node. A cluster that uses this security configuration must specify the realm name and KDC admin password of the external KDC.
`"ClusterDedicatedKdcConfiguration": {`	Required when *ClusterDedicatedKdc* is specified.
`"TicketLifetimeInHours": 24,`	Optional. Specifies the period for which a Kerberos ticket issued by the KDC is valid on clusters that use this security configuration. 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 `kinit` from the master node command line to renew after a ticket expires.
`"CrossRealmTrustConfiguration": {`	Specifies a cross-realm trust between a cluster-dedicated KDC on clusters that use this security configuration and a KDC in a different Kerberos realm. Principals (typically users) from another realm are authenticated to clusters that use this configuration. Additional configuration in the other Kerberos realm is required. For more information, see Tutorial: Configure a cross-realm trust with an Active Directory domain (p. 468).
`"Realm": "KDC2.COM",`	Specifies the Kerberos realm name of the other realm in the trust relationship. By convention, Kerberos realm names are the same as the domain name but in all capital letters.
`"Domain": "kdc2.com",`	Specifies the domain name of the other realm in the trust relationship.

Parameter	Description
`"AdminServer": "kdc.com:749",`	Specifies the fully qualified domain name (FQDN) or IP address 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`).
`"KdcServer": "kdc.com:88"`	Specifies the fully qualified domain name (FQDN) or IP address 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 use 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`).
`}`	
`}`	
"ExternalKdcConfiguration": {	Required when *`ExternalKdc`* is specified.
`"TicketLifetimeInHours": 24,`	Optional. Specifies the period for which a Kerberos ticket issued by the KDC is valid on clusters that use this security configuration. 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 `kinit` from the master node command line to renew after a ticket expires.
`"KdcServerType": "Single",`	Specifies that a single KDC server is referenced. `Single` is currently the only supported value.

Parameter	Description
"AdminServer": "*kdc.com:749*",	Specifies the fully qualified domain name (FQDN) or IP address of the external admin server. 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`).
"KdcServer": "*kdc.com:88*",	Specifies the fully qualified domain name (FQDN) of the external KDC server. The KDC server and admin server typically run on the same machine with the same FQDN, but use 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`).
"AdIntegrationConfiguration": {	Specifies that Kerberos principal authentication is integrated with a Microsoft Active Directory domain.
"AdRealm": "*AD.DOMAIN.COM*",	Specifies the Kerberos realm name of the Active Directory domain. By convention, Kerberos realm names are typically the same as the domain name but in all capital letters.
"AdDomain": "*ad.domain.com*"	Specifies the Active Directory domain name.
"AdServer": "*ad.domain.com*"	Specifies the fully qualified domain name (FQDN) of the Microsoft Active Directory domain controller.
}	
}	
}	
}	

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. 420).

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 `arn:aws:iam::`*`account-id`*`:role/`*`role-name`*. This is the IAM role that Amazon EMR assumes if the EMRFS request to Amazon S3 matches any of the `Identifiers` specified.
`"IdentifierType":`	Can be one of the following: • `"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

Parameter	Description
	prefix `s3://mybucket/` matches `s3://mybucket/mydir` and `s3://mybucket/yetanotherdir`. • `"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.

Configure metadata service requests to Amazon EC2 instances

Instance metadata is data about your instance that you can use to configure or manage the running instance. You can access instance metadata from a running instance using one of the following methods:

- Instance Metadata Service Version 1 (IMDSv1) - a request/response method
- Instance Metadata Service Version 2 (IMDSv2) - a session-oriented method

While Amazon EC2 supports both IMDSv1 and IMDSv2, Amazon EMR supports IMDSv2 in Amazon EMR 5.23.1, 5.27.1, 5.32 or later, and 6.2 or later. In these releases, Amazon EMR components use IMDSv2 for all IMDS calls. For IMDS calls in your application code, you can use both IMDSv1 and IMDSv2, or configure the IMDS to use only IMDSv2 for added security. When you specify that IMDSv2 must be used, IMDSv1 no longer works.

For more information, see Configure the instance metadata service in the *Amazon EC2 User Guide for Linux Instances*.

> **Note**
> In earlier Amazon EMR 5.x or 6.x releases, turning off IMDSv1 causes cluster startup failure as Amazon EMR components use IMDSv1 for all IMDS calls. When turning off IMDSv1, please ensure that any custom software that utilizes IMDSv1 is updated to IMDSv2.

Specifying instance metadata service configuration using the AWS CLI

The following is an example JSON snippet for specifying Amazon EC2 instance metadata service (IMDS) within a security configuration.

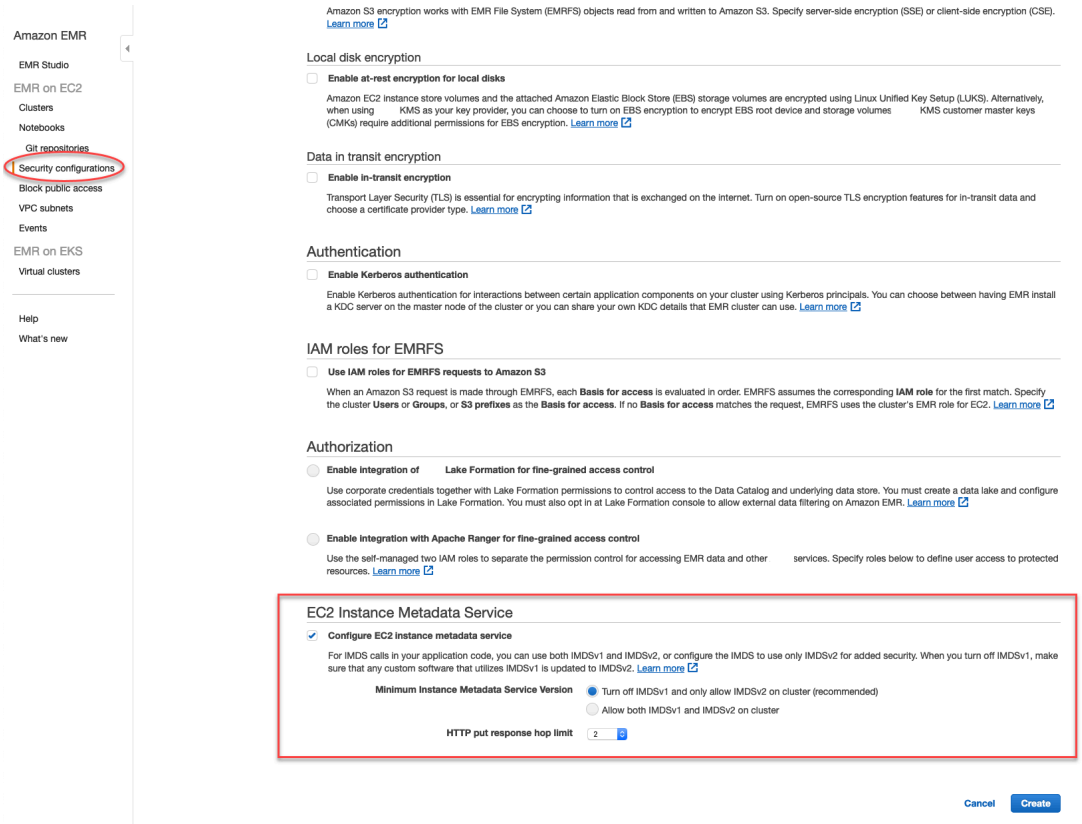
```
{
  "InstanceMetadataServiceConfiguration" : {
      "MinimumInstanceMetadataServiceVersion": integer,
      "HttpPutResponseHopLimit": integer
   }
}
```

Parameter	Description
`"InstanceMetadataServiceConfiguration":`	Required.
`"MinimumInstanceMetadataServiceVersion":`	Required. Specify 1 or 2. A value of 1 allows IMDSv1 and IMDSv2. A value of 2 allows only IMDSv2.

Parameter	Description
`"HttpPutResponseHopLimit":`	Required. The desired HTTP PUT response hop limit for instance metadata requests. The larger the number, the further instance metadata requests can travel. Default: 1. Specify an integer from 1 to 64.

Specifying instance metadata service configuration using the console

You can configure the use of IMDS for a cluster when you launch it from the Amazon EMR console.



To configure the use of IMDS using the console:

1. When creating a new security configuration on the **Security configurations** page, select **Configure EC2 Instance metadata service** under the **EC2 Instance Metadata Service** setting. This configuration is supported only in Amazon EMR 5.23.1, 5.27.1, 5.32 or later, and 6.2 or later.
2. For the **Minimum Instance Metadata Service Version** option, select either:
   - **Turn off IMDSv1 and only allow IMDSv2**, if you want to allow only IMDSv2 on this cluster. See Transition to using instance metadata service version 2 in the *Amazon EC2 User Guide for Linux Instances*.
   - **Allow both IMDSv1 and IMDSv2 on cluster**, if you want to allow IMDSv1 and session-oriented IMDSv2 on this cluster.
3. For IMDSv2, you can also configure the allowable number of network hops for the metadata token by setting the **HTTP put response hop limit** to an integer between 1 and 64.

For more information, see Configure the instance metadata service in the *Amazon EC2 User Guide for Linux Instances*.

See Configure instance details and Configure the instance metadata service in the *Amazon EC2 User Guide for Linux Instances*.

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 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/emr.
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 m5.xlarge --release-label emr-5.0.0 --security-
configuration mySecConfig
```

Data protection in Amazon EMR

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

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

- Use multi-factor authentication (MFA) with each account.
- Use TLS to communicate with AWS resources. We require TLS 1.2.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a **Name** field. This includes when you work with Amazon EMR or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into Amazon EMR or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

Encrypt data at rest and in transit

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) 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, which is not configured using security configurations. For more information, see Transparent encryption in HDFS on Amazon EMR in the *Amazon EMR Release Guide*.

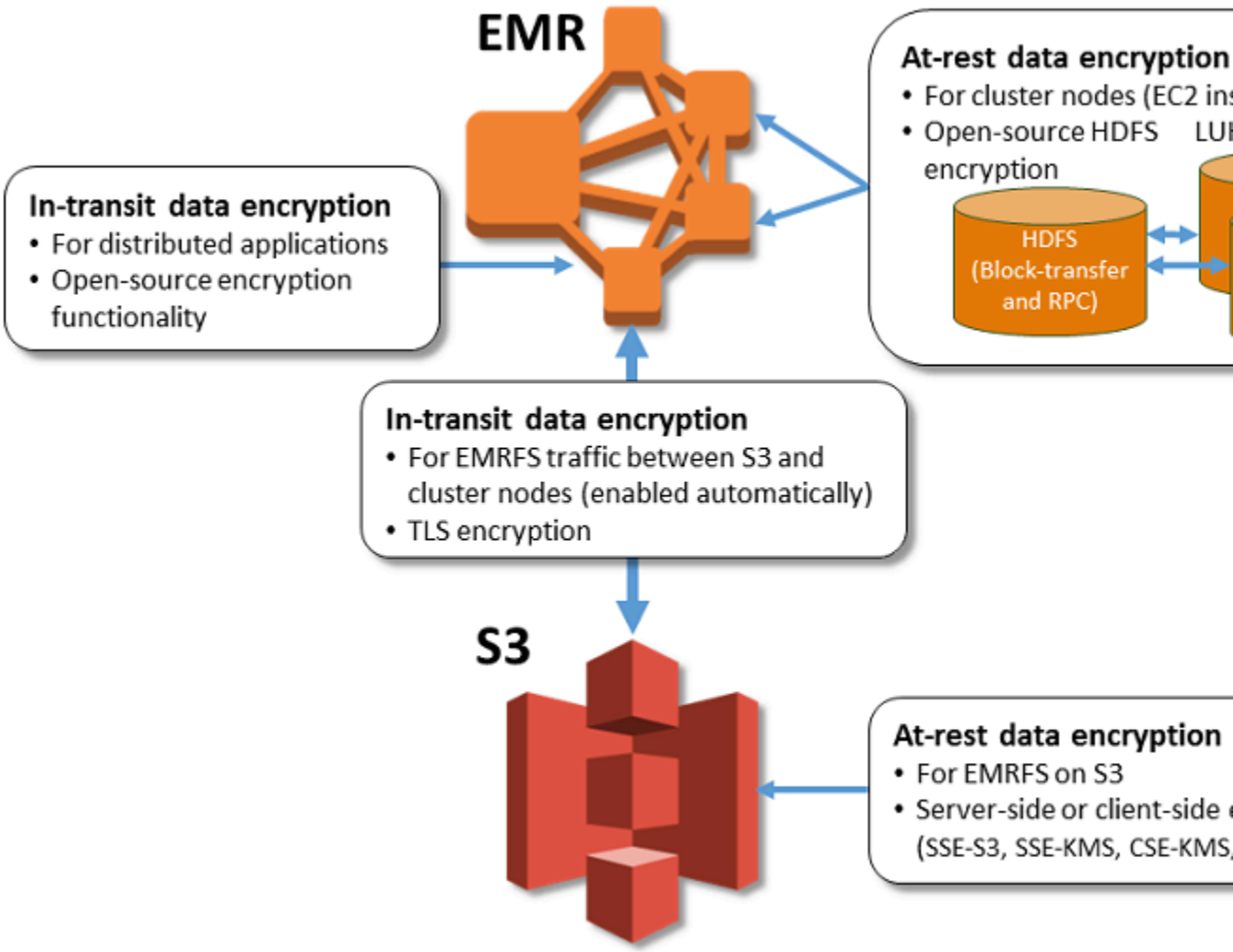
Topics

- Encryption options (p. 374)
- Create keys and certificates for data encryption (p. 378)

Encryption options

With Amazon EMR versions 4.8.0 and later, you can use a security configuration to specify settings for encrypting data at rest, data in transit, or both. When you enable at-rest data encryption, you can choose to encrypt EMRFS data in Amazon S3, data in local disks, 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. 355).

The following diagram shows the different data encryption options available with security configurations.



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

- Optionally, with Amazon EMR versions 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*.
- 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.
- If you are using an Amazon EMR version earlier than 5.24.0, an encrypted EBS root device 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 in the *Amazon EMR Management Guide*.

> **Note**
> Beginning with Amazon EMR version 5.24.0, you can use a security configuration option to encrypt EBS root device and storage volumes when you specify AWS KMS as your key provider. For more information, see Local disk encryption (p. 376).

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.

Encryption at rest 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) as the **Default encryption mode** when you enable encryption at rest. Optionally, you can specify different encryption methods for individual buckets using **Per bucket encryption overrides**. Regardless of whether Amazon S3 encryption is enabled, Transport Layer Security (TLS) encrypts the EMRFS objects in transit between 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 User Guide*.

> **Note**
> When you use AWS KMS, charges apply for the storage and use of encryption keys. For more information, see AWS KMS Pricing.

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 User 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 key to set up with policies suitable for Amazon EMR. For more information about key requirements for Amazon EMR, see Using AWS KMS keys for encryption.

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 root 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 User 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.

Local disk encryption

The following mechanisms work together to encrypt local disks when you enable local disk encryption using an Amazon EMR security configuration.

Open-source HDFS encryption

HDFS exchanges data between cluster instances during distributed processing. It 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. For more information, see Encryption in transit (p. 377). 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*.

Instance store encryption

For EC2 instance types that use NVMe-based SSDs as the instance store volume, NVMe encryption is used regardless of Amazon EMR encryption settings. For more information, see NVMe SSD volumes in the *Amazon EC2 User Guide for Linux Instances*. For other instance store volumes, Amazon EMR uses LUKS to encrypt the instance store volume when local disk encryption is enabled regardless of whether EBS volumes are encrypted using EBS encryption or LUKS.

EBS volume encryption

If you create a cluster in a Region where Amazon EC2 encryption of EBS volumes is enabled by default for your account, EBS volumes are encrypted even if local disk encryption is not enabled. For more information, see Encryption by default in the *Amazon EC2 User Guide for Linux Instances*. With local disk encryption enabled in a security configuration, the Amazon EMR settings take precedence over the Amazon EC2 encryption-by-default settings for cluster EC2 instances.

The following options are available to encrypt EBS volumes using a security configuration:

- **EBS encryption** – Beginning with Amazon EMR version 5.24.0, you can choose to enable EBS encryption. The EBS encryption option encrypts the EBS root device volume and attached storage volumes. The EBS encryption option is available only when you specify AWS Key Management Service as your key provider. We recommend using EBS encryption.
- **LUKS encryption** – If you choose to use LUKS encryption for Amazon EBS volumes, the LUKS encryption applies only to attached storage volumes, not to the root device volume. For more information about LUKS encryption, see the LUKS on-disk specification.

  For your key provider, you can set up an AWS KMS key 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.

> **Note**
> To check if EBS encryption is enabled on your cluster, it is recommended that you use `DescribeVolumes` API call. For more information, see DescribeVolumes. Running `lsblk` on the cluster will only check the status of LUKS encryption, instead of EBS encryption.

Encryption in transit

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 application configurations. For more information, see Configure applications.

- **Apache Hadoop**
  - 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).
  - For more information, see Hadoop in secure mode in the Apache Hadoop documentation.
- **Apache 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 the Apache HBase documentation.
  - For more information about Kerberos with Amazon EMR, see Use Kerberos authentication (p. 448).
- **Apache Hive**
  - JDBC/ODBC client communication with HiveServer2 is encrypted using SSL configurations in Amazon EMR releases 6.9.0 and later.
  - For more information, see the SSL encryption section of the Apache Hive documentation.
- **Apache Spark**
  - Internal RPC communication between Spark components, such as the block transfer service and the external shuffle service, is encrypted using the AES-256 cipher in Amazon EMR versions 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.
  - For more information, see Spark security settings section of the Apache Spark documentation.
- **Apache Tez**
  - Tez shuffle handler uses TLS (`tez.runtime.ssl.enable`).
- **Presto**
  - Internal communication between Presto nodes uses SSL/TLS (Amazon EMR version 5.6.0 and later only).

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 encrypting data in transit with Amazon EMR encryption (p. 381).

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, you can use AWS KMS or a custom provider that you create. Next, create the keys or key provider as described in this section.

Providing keys for encrypting data at rest 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 a KMS key 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 keys 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 KMS key 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 KMS key. For more information, see Using Key Policies in the *AWS Key Management Service Developer Guide* and Configure IAM roles for EMRFS requests to Amazon S3.

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 KMS key, or you can use the AWS CLI or an AWS SDK to attach an appropriate key policy.

Note that Amazon EMR supports only symmetric KMS keys. You cannot use an asymmetric KMS key to encrypt data at rest in an Amazon EMR cluster. For help determining whether a KMS key is symmetric or asymmetric, see Identifying symmetric and asymmetric KMS keys.

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 KMS key. To create a new KMS key, 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 KMS key to modify.
4. On the key details page under **Key Users**, choose **Add**.
5. In the **Add key users** dialog box, select the appropriate role. The name of the default role is `EMR_EC2_DefaultRole`.
6. Choose **Add**.

Enabling EBS encryption by providing additional permissions for KMS keys

Beginning with Amazon EMR version 5.24.0, you can encrypt EBS root device and storage volumes by using a security configuration option. To enable such option, you must specify AWS KMS as your key provider. Additionally, you must grant the EMR service role `EMR_DefaultRole` with permissions to use the AWS KMS key that you specify.

You can use the AWS Management Console to add the EMR service role to the list of key users for the specified KMS key, 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 service role, `EMR_DefaultRole` as a *key user* using the AWS Management Console. It assumes that you have already created a KMS key. To create a new KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*.

To add the EMR service role 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. Choose **Customer managed keys** in the left sidebar.
4. Select the alias of the KMS key to modify.

5. On the key details page under **Key Users**, choose **Add**.
6. In the **Add key users** dialog box, select the appropriate role. The name of the default EMR service role is `EMR_DefaultRole`.
7. Choose **Add**.

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 encrypting data in transit with Amazon EMR encryption

With Amazon EMR release version 4.8.0 or later, you have two options for specifying artifacts for encrypting data in transit using a security configuration:

- 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 Encryption in transit (p. 377).

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
trustedCertificates.pem	Optional	Required if the provided certificate is not signed by either the Java default trusted root

File name	Required/optional	Details
		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 `jre/lib/security/cacerts`.

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`.

If the provided PEM file in the encryption artifact doesn't have a wildcard character in the CN for the domain, you must change the value of `hadoop.ssl.hostname.verifier` to `ALLOW_ALL`. This is done with the `core-site` classification when submitting configurations to a cluster or by adding this value in the `core-site.xml` file. This change is required because the default hostname verifier won't accept a hostname without the wildcard, resulting in an error. For more information about EMR cluster configuration within an Amazon VPC, see Configure networking (p. 306).

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), and Locale (L), 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
```

AWS Identity and Access Management for Amazon EMR

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Amazon EMR resources. IAM is an AWS service that you can use with no additional charge.

Topics

- Audience (p. 383)

- Authenticating with identities (p. 383)
- Managing access using policies (p. 385)
- How Amazon EMR works with IAM (p. 386)
- Configure runtime roles for Amazon EMR steps (p. 388)
- Configure IAM service roles for Amazon EMR permissions to AWS services and resources (p. 394)
- Amazon EMR identity-based policy examples (p. 426)

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work you do in Amazon EMR.

Service user – If you use the Amazon EMR service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Amazon EMR features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator.

Service administrator – If you're in charge of Amazon EMR resources at your company, you probably have full access to Amazon EMR. It's your job to determine which Amazon EMR features and resources your employees should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with Amazon EMR, see How Amazon EMR works with IAM (p. 386).

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Amazon EMR. To view example Amazon EMR identity-based policies that you can use in IAM, see Amazon EMR identity-based policy examples (p. 426).

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. For more information about signing in using the AWS Management Console, see The IAM console and sign-in page in the *IAM User Guide*.

You must be *authenticated* (signed in to AWS) as the AWS account root user, an IAM user, or by assuming an IAM role. You can also use your company's single sign-on authentication, or even sign in using Google or Facebook. In these cases, your administrator previously set up identity federation using IAM roles. When you access AWS using credentials from another company, you are assuming a role indirectly.

To sign in directly to the AWS Management Console, use your password with your root user email or your IAM user name. You can access AWS programmatically using your root user or IAM user access keys. AWS provides SDK and command line tools to cryptographically sign your request using your credentials. If you don't use AWS tools, you must sign the request yourself. Do this using *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 signing process in the *AWS General Reference*.

Regardless of the authentication method that you use, you might also be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is

accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *AWS General Reference*.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. An IAM user can have long-term credentials such as a user name and password or a set of access keys. To learn how to generate access keys, see Managing access keys for IAM users in the *IAM User Guide*. When you generate access keys for an IAM user, make sure you view and securely save the key pair. You cannot recover the secret access key in the future. Instead, you must generate a new access key pair.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to create an IAM user (instead of a role) in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM roles in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Temporary IAM user permissions** – An IAM user can assume an IAM role to temporarily take on different permissions for a specific task.
- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Creating a role for a third-party Identity Provider in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center (successor to AWS Single Sign-On) User Guide*.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.
- **AWS service access** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2

instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

To learn whether to use IAM roles, see When to create an IAM role (instead of a user) in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to IAM identities or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when an entity (root user, IAM user, or IAM role) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

An IAM administrator can use policies to specify who has access to AWS resources, and what actions they can perform on those resources. Every IAM entity (user or role) starts with no permissions. In other words, by default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or the administrator can add the user to a group that has the intended permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, role, or group. These policies control what actions that identity can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing between managed policies and inline policies in the *IAM User Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all

features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs work in the *AWS Organizations User Guide*.

- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How Amazon EMR works with IAM

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. Amazon EMR supports specific actions, resources, and condition keys. To learn about all of the elements that you use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Amazon EMR does not support resource-based policies.

Actions

The `Action` element of an IAM identity-based policy describes the specific action or actions that will be allowed or denied by the policy. Policy actions usually have the same name as the associated AWS API operation. The action is used in a policy to grant permissions to perform the associated operation.

Policy actions in Amazon EMR use the following prefix before the action: `elasticmapreduce:`. For example, to grant someone permission to create a cluster using the `RunJobFlow` API operation, you include the `elasticmapreduce:RunJobFlow` action in their policy. Policy statements must include either an `Action` or `NotAction` element. Amazon EMR defines its own set of actions that describe tasks that you can perform with this service.

To specify multiple actions in a single statement, separate them with commas as follows:

```
"Action": [
      "elasticmapreduce:action1",
      "elasticmapreduce:action2"
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `Describe`, include the following action:

```
"Action": "elasticmapreduce:Describe*"
```

To see a list of Amazon EMR actions, see Actions Defined by Amazon EMR in the *IAM User Guide*.

Resources

The `Resource` element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. You specify a resource using an ARN or using the wildcard (*) to indicate that the statement applies to all resources.

To see a list of Amazon EMR resource types and their ARNs, see Resources Defined by Amazon EMR in the *IAM User Guide*. To learn with which actions you can specify the ARN of each resource, see Actions Defined by Amazon EMR.

Condition keys

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can build conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: Variables and tags in the *IAM User Guide*.

Amazon EMR defines its own set of condition keys and also supports using some global condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

All Amazon EC2 actions support the `aws:RequestedRegion` and `ec2:Region` condition keys. For more information, see Example: Restricting access to a specific Region.

To see a list of Amazon EMR condition keys, see Condition Keys for Amazon EMR in the *IAM User Guide*. To learn with which actions and resources you can use a condition key, see Actions Defined by Amazon EMR.

Use cluster and Notebook tags with IAM policies for access control

Permission for Amazon EMR actions associated with EMR Notebooks and EMR clusters can be fine-tuned using tag-based access control with identity-based IAM policies. You can use *condition keys* within 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 (which creates an EMR notebook) and the `RunJobFlow` action (which creates a cluster) so that a request for a tag must be submitted when the resource is created.

In Amazon EMR, the condition keys that can be used in a `Condition` element apply only to those Amazon EMR API actions where `ClusterID` or `NotebookID` is a required request parameter. For example, the ModifyInstanceGroups action does not support context keys because `ClusterID` is an optional parameter.

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 following condition keys are available in Amazon EMR:

- Use the `elasticmapreduce:ResourceTag/`*`TagKeyString`* condition context key to allow or deny user actions on clusters or notebooks with tags that have the *`TagKeyString`* that you specify. If an action passes both `ClusterID` and `NotebookID`, 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 that you specify. 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 Amazon EMR identity-based policy examples (p. 426).
- Use the `elasticmapreduce:RequestTag/`*`TagKeyString`* condition context key to require a specific tag with actions/API calls. For example, you can use this condition context key along with the

`CreateEditor` action to require that a key with *`TagKeyString`* is applied to a notebook when it is created.

Examples

To view examples of Amazon EMR identity-based policies, see Amazon EMR identity-based policy examples (p. 426).

Configure runtime roles for Amazon EMR steps

A *runtime role* is an AWS Identity and Access Management (IAM) role that you can specify with Amazon EMR steps. Jobs that you submit with Amazon EMR steps use the runtime role to access AWS resources, such as objects in Amazon S3. You can also enforce table and column permissions based on your Amazon EMR runtime role to manage your access to data lakes with AWS Lake Formation. Apache Hive and Apache Spark jobs use these permissions at runtime. This way, you can manage access control for each Amazon EMR step individually, instead of all steps sharing the Amazon EMR cluster's instance profile.

> **Note**
> When you specify a runtime role for an Amazon EMR step, jobs that you submit with Amazon EMR steps can only access AWS resources that the policy attached to the runtime role allows. These jobs can't use the cluster's EC2 instance profile to access any AWS resources.

Prerequisites for launching an Amazon EMR cluster with a runtime role

Step 1: Set up security configurations in Amazon EMR

Use the following JSON structure to create a security configuration, and set `EnableApplicationScopedIAMRole` to `true`.

```
{
    "AuthorizationConfiguration":{
        "IAMConfiguration":{
            "EnableApplicationScopedIAMRole":true
        }
    }
}
```

When you use a runtime role with Amazon EMR steps, you don't need to enable Kerberos in the security configuration. Amazon EMR automatically configures your clusters to use MIT KDC locally and manages the principals in the MIT KDC that run on the cluster.

`PropagateSourceIdentity`

When you use a source identity, you can monitor and control actions that you take with IAM roles. If you already use source identity, set `ApplicationScopedIAMRoleConfiguration/PropagateSourceIdentity` to `true` to activate the source identity, as follows.

```
{
    "AuthorizationConfiguration":{
        "IAMConfiguration":{
            "EnableApplicationScopedIAMRole":true,
            "ApplicationScopedIAMRoleConfiguration":{
                "PropagateSourceIdentity":true
            }
        }
```

```
    }
}
```

When you set `PropagateSourceIdentity` to `true`, Amazon EMR applies the source identity from the session credentials used to call the Amazon EMR `AddJobFlowSteps` API to a session you create with the runtime role. If no source identity is present in the session credentials, then Amazon EMR won't set the source identity on the session created using the runtime role. To use this property, you must provide `sts:SetSourceIdentity` permissions to the instance profile and ensure that your runtime role trusts your instance profile to set source identity. To learn more about source identity, see Monitor and control actions taken with assumed roles.

`LakeFormationConfiguration`

You can use Lake Formation to apply fine-grained access control (FGAC) for your resources in AWS Glue Data Catalog and data in Amazon S3. If you use Lake Formation, you can set the `LakeFormationConfiguration/AuthorizedSessionTagValue` property to `Amazon EMR` to ensure that those permissions apply to jobs submitted with Amazon EMR steps, as follows.

```
{
    "AuthorizationConfiguration":{
        "IAMConfiguration":{
            "EnableApplicationScopedIAMRole":true,
            "LakeFormationConfiguration":{
                "AuthorizedSessionTagValue":"Amazon EMR"
            }
        }
    }
}
```

> **Note**
> Amazon EMR only supports access control based on Lake Formation with Apache Hive and Apache Spark.

After you create your security configuration, you can launch your cluster. For instructions on how to launch your cluster, see Specify a security configuration for a cluster (p. 373).

Step 2: Set up an Amazon EMR runtime role

Attach IAM policies to your Amazon EMR runtime role so that your jobs can access other AWS resources. To add or edit policies to an IAM role, see Adding and removing IAM identity permissions and Editing IAM policies.

If you want to ensure that Lake Formation can access your data lake, you can attach additional policies so that Amazon EMR can access Lake Formation and the AWS Glue Data Catalog. The following example shows how to allow your runtime role to create a table, update table metadata, and read table metadata. You can customize these policies as needed.

```
{
    "Version":"2012-10-17",
    "Statement":{
        "Sid":"LakeFormationManagedAccess",
        "Effect":"Allow",
        "Action":[
            "lakeformation:GetDataAccess",
            "glue:Get*",
            "glue:Create*",
            "glue:Update*"
        ],
        "Resource":"*"
    }
```

```
}
```

To define Lake Formation permissions for this role, see Step 4: Set up a trust policy (p. 390). When you turn on Lake Formation, make sure to grant only the Amazon EMR runtime role access to S3 buckets that aren't associated with your data lake. Provide access to your data lake S3 buckets only through Lake Formation and AWS Glue Data Catalog. Don't explicitly grant S3 bucket access.

Step 3: Set up an EC2 instance profile for the Amazon EMR cluster

To use runtime roles with Amazon EMR steps, add the following policies to the IAM role that you plan to use as the instance profile role. With these policies, the underlying EC2 instances for the Amazon EMR cluster can assume the runtime role and apply a tag to that runtime role. You only need the `PropagateSourceIdentity` statement if you turn on source identity propagation in the security configuration that you set up in Step 1: Set up security configurations in Amazon EMR (p. 388).

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"AllowRuntimeRoleUsage",
            "Effect":"Allow",
            "Action":[
                "sts:AssumeRole",
                "sts:TagSession"
            ],
            "Resource":[
                <runtime-role-ARN>
            ]
        },
        { // If propagating source identity
            "Sid":"PropagateSourceIdentity",
            "Effect":"Allow",
            "Action":"sts:SetSourceIdentity",
            "Resource":[
                <runtime-role-ARN>
            ],
            "Condition":{
                "StringEquals":{
                    "sts:SourceIdentity":<source-identity>
                }
            }
        }
    ]
}
```

To add policies to an IAM role or edit an existing inline or managed policy, see Adding and removing IAM identity permissions.

Step 4: Set up a trust policy

For each IAM role that you plan to use as a runtime role, update the trust relationship so that the EC2 instance profile role can assume the runtime role. The following example policy instance profile applies the `LakeFormationAuthorizedCaller` tag for the instance profile `EMR_EC2_DefaultRole`. You only need the `AllowSetSourceIdentity` statement if you turned on source identity propagation in the security configuration that you set up in Step 1: Set up security configurations in Amazon EMR (p. 388). You need a `AllowPassSessionTagsAndTransitive` statement only if you activated Lake Formation in your security configuration. To modify an IAM role's trust policy, see Modifying a role trust policy.

```
{
    "Sid":"AllowAssumeRole",
    "Effect":"Allow",
```

```
    "Principal":{
        "AWS":"arn:aws:iam::<AWS_ACCOUNT_ID>:role/EMR_EC2_DefaultRole"
    },
    "Action":"sts:AssumeRole"
},
{
    "Sid":"AllowPassSessionTagsAndTransitive",
    "Effect":"Allow",
    "Principal":{
        "AWS":"arn:aws:iam::<AWS_ACCOUNT_ID>:role/EMR_EC2_DefaultRole"
    },
    "Action":"sts:TagSession"
},
{
    "Sid":"AllowSetSourceIdentity",
    "Effect":"Allow",
    "Principal":{
        "AWS":"arn:aws:iam::<AWS_ACCOUNT_ID>:role/EMR_EC2_DefaultRole"
    },
    "Action":[
        "sts:SetSourceIdentity",
        "sts:AssumeRole"
    ],
    "Condition":{
        "StringEquals":{
            "sts:SourceIdentity":<source-identity>
        }
    }
}
```

(Optional) Step 5: Set up Amazon EMR clusters to use Lake Formation

Use this step if you use Lake Formation to access your data lake. You need this step for Amazon EMR jobs to access Lake Formation. Make sure that you use `Amazon EMR` in the `AuthorizedSessionTagValueList`, as the following example shows.

```
aws lakeformation put-data-lake-settings --cli-input-json file://datalakesettings.json
```

```
{
    "DataLakeSettings":{
        "DataLakeAdmins":[
            {
                "DataLakePrincipalIdentifier":"arn:aws:iam::<AWS_ACCOUNT_ID>:user/
lakeAdmin"
            }
        ],
        "CreateDatabaseDefaultPermissions":[

        ],
        "CreateTableDefaultPermissions":[

        ],
        "TrustedResourceOwners":[

        ],
        "AllowExternalDataFiltering":true,
        "ExternalDataFilteringAllowList":[
            {
                "DataLakePrincipalIdentifier":<AWS_ACCOUNT_ID>
            }
        ],
        "AuthorizedSessionTagValueList":[
            "Amazon EMR"
```

```
        ]
    }
}
```

Launch an Amazon EMR cluster with role-based access control

After you set up your configurations, you can launch an Amazon EMR cluster with the security configuration from Step 1: Set up security configurations in Amazon EMR (p. 388). Use release label `emr-6.7.0` or later, and select Hive, Spark, or both as your cluster application. For instructions on how to launch your cluster, see Specify a security configuration for a cluster (p. 373).

Submit Spark jobs using Amazon EMR steps

The following is an example of how to run the HdfsTest example included with Apache Spark. This API call only succeeds if the provided Amazon EMR runtime role can access the `S3_LOCATION`.

```
RUNTIME_ROLE_ARN=<runtime-role-arn>
S3_LOCATION=<s3-path>
REGION=<aws-region>
CLUSTER_ID=<cluster-id>

aws emr add-steps --cluster-id $CLUSTER_ID \
--steps '[{ "Name": "Spark Example", "ActionOnFailure": "CONTINUE","HadoopJarStep":
 { "Jar":"command-runner.jar","Args" : ["spark-example","HdfsTest", "$S3_LOCATION"] } }]' \
--execution-role-arn $RUNTIME_ROLE_ARN \
--region $REGION
```

Note
We recommend that you turn off SSH access to the Amazon EMR cluster and only allow the Amazon EMR `AddJobFlowSteps` API to access to the cluster.

Submit Hive jobs using Amazon EMR steps

The following example uses Apache Hive with EMR steps to submit a job to run the `QUERY_FILE.hql` file. This query only succeeds if the provided runtime role can access the S3 path of the query file.

```
RUNTIME_ROLE_ARN=<runtime-role-arn>
REGION=<aws-region>
CLUSTER_ID=<cluster-id>

aws emr add-steps --cluster-id $CLUSTER_ID \
--steps '[{ "Name": "Run hive query using command-runner.jar - simple
 select","ActionOnFailure":"CONTINUE","HadoopJarStep": { "Jar": "command-
runner.jar","Args" :["hive -
f","s3://DOC_EXAMPLE_BUCKET/QUERY_FILE.hql"] } }]' \
--execution-role-arn $RUNTIME_ROLE_ARN \
--region $REGION
```

Control access to the Amazon EMR runtime role

You can control access to the runtime role with the condition key `elasticmapreduce:ExecutionRoleArn`. The following policy allows an IAM principal to use an IAM role named `Caller`, or any IAM role that begins with the string `CallerTeamRole`, as the runtime role with Amazon EMR steps.

Important
You must create a condition based on the `elasticmapreduce:ExecutionRoleArn` context key when you grant a caller access to call `AddJobFlowSteps` API, as the following example shows.

```
{
    "Sid":"AddStepsWithSpecificExecRoleArn",
    "Effect":"Allow",
    "Action":[
        "elasticmapreduce:AddJobFlowSteps"
    ],
    "Resource":"*",
    "Condition":{
        "StringEquals":{
            "elasticmapreduce:ExecutionRoleArn":[
                "arn:aws:iam::<AWS_ACCOUNT_ID>:role/Caller"
            ]
        },
        "StringLike":{
            "elasticmapreduce:ExecutionRoleArn":[
                "arn:aws:iam::<AWS_ACCOUNT_ID>:role/CallerTeamRole*"
            ]
        }
    }
}
```

Use an external ID with a runtime role

When you assume the runtime role with an external ID, you can assert the circumstances where you're operating. You can use an external ID with a runtime role to assert that only Amazon EMR clusters with a specific security configuration attached can assume the IAM runtime role with Amazon EMR steps.

Amazon EMR generates a unique identifier `ExternalId` for each security configuration that has runtime roles for the activated Amazon EMR steps. You can find the external ID with the Amazon EMR `DescribeSecurityConfiguration` API, as shown in the following example. All the `AssumeRole` calls for the runtime role on the cluster use this external ID. You can optionally change the trust policy on the runtime role with this external ID.

```
aws emr describe-security-configuration --name 'iamconfig-with-lf'{"Name": "iamconfig-with-
lf",
    "SecurityConfiguration":
        "{\"AuthorizationConfiguration\":{\"IAMConfiguration\":
{\"EnableApplicationScopedIAMRole\
        ":true,\"ApplicationScopedIAMRoleConfiguration\":{\"PropagateSourceIdentity\":true,
\"Exter
        nalId\":\"FXH5TSACFDWUCDSR3YQE2O7ETPUSM4OBCGLYWODSCUZDNZ4Y\"}},\"Lake
        FormationConfiguration\":{\"AuthorizedSessionTagValue\":\"Amazon EMR\"}}}",
    "CreationDateTime": "2022-06-03T12:52:35.308000-07:00"
}
```

Additional considerations

Additionally, consider the following when you configure runtime roles for Amazon EMR.

- Amazon EMR steps only supports commercial AWS Regions.
- You must use Amazon EMR release version `emr-6.7.0` or higher.
- Amazon EMR steps supports Apache Spark and Apache Hive (with Apache Tez).
- You can only specify runtime roles with jobs that you submit with Amazon EMR steps.
- Currently, we only support steps that use `AddJobFlowSteps`. We don't support steps that use `RunJobFlow`.
- Amazon EMR steps supports Livy endpoints and JDBC/ODBC clients.
- Operations that support access controls based on Lake Formation with Apache Spark and Apache Hive include `select`, `describe`, `show database`, `show table`, `show column`, and `show partition`.

- Operations that support access controls based on Lake Formation with non-goverened Apache Spark tables include `insert into`, `insert overwrite`, and `alter table`.
- Amazon EMR doesn't support access controls based on Lake Formation. These include the following:
  - Writes to governed tables
  - Writes to non-governed Apache Hive tables
  - Use of the Lake Formation data filter
  - Use of any DDLs
- Amazon EMR steps doesn't support high availability clusters.
- Amazon EMR steps doesn't support Apache Ranger.

Configure IAM service roles for Amazon EMR permissions to AWS services and resources

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*. For more information, see IAM roles and Using instance profiles in the *IAM User Guide*. The IAM policies attached to these roles provide permissions for the cluster to interoperate with other AWS services on behalf of a user.

An additional role, the Auto Scaling role, is required if your cluster uses automatic scaling in Amazon EMR. The AWS service role for EMR Notebooks is required if you use EMR Notebooks.

Amazon EMR provides default roles and default managed policies that determine permissions for each role. Managed policies are created and maintained by AWS, so they are updated automatically if service requirements change. See AWS managed policies in the *IAM User Guide*.

If you are creating a cluster or notebook for the first time in an account, roles for Amazon EMR do not yet exist. 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/). You can specify default roles for Amazon EMR to create and use, you can create your own roles and specify them individually when you create a cluster to customize permissions, and you can specify default roles to be used when you create a cluster using the AWS CLI. For more information, see Customize IAM roles (p. 418).

Modifying identity-based policies for permissions to pass service roles for Amazon EMR

The Amazon EMR full permissions default managed policy (`AmazonEMRFullAccessPolicy_v2`) and Amazon EMR service policy (`AmazonEMRServicePolicy_v2`) are available to replace the soon-to-be-deprecated policy. The v2 policies incorporate new iam:PassRole security configurations, including the following:

- `iam:PassRole` permissions only for specific default Amazon EMR roles.
- `iam:PassedToService` conditions that allow you to use the policy with only specified AWS services, such as `elasticmapreduce.amazonaws.com` and `ec2.amazonaws.com`.

You can view the JSON version of the AmazonEMRFullAccessPolicy_v2 and AmazonEMRServicePolicy_v2 policies in the IAM console.

We recommend that you create new clusters using v2 managed policies.

Service role summary

The following table lists the IAM service roles associated with Amazon EMR for quick reference.

Function	Default role	Description	Default managed policy
Service role for Amazon EMR (EMR role) (p. 398)	`EMR_DefaultRole`	Allows Amazon EMR to call other AWS services on your behalf when provisioning resources and performing service-level actions. This role is required for all clusters.	`AmazonElasticMapReduceRole` **Important** Requesting Spot Instances requires a service-linked role. If this role doesn't exist, the EMR service role must have permission to create it or a permission error occurs. If you plan to request Spot Instances, you must update this policy to include a statement that allows the creation of this service-linked role. For more information, see Service role for Amazon EMR (EMR role) (p. 398) and Service-linked role for Spot Instance requests in the *Amazon EC2 User Guide for Linux Instances*.
Service role for cluster EC2 instances (EC2 instance profile) (p. 403)	`EMR_EC2_DefaultRole`	Application processes that run on top of the Hadoop ecosystem on cluster instances use this role when they call other AWS services. For accessing data in Amazon S3 using EMRFS, you can	`AmazonElasticMapReduceforEC2R` For more information, see Service role for cluster EC2 instances (EC2 instance profile) (p. 403).

Function	Default role	Description	Default managed policy
		specify different roles to be assumed based on the location of data in Amazon S3. For example, multiple teams can access a single Amazon S3 data "storage account." For more information, see Configure IAM roles for EMRFS requests to Amazon S3 (p. 420). This role is required for all clusters.	
Service role for automatic scaling in EMR (Auto Scaling role) (p. 408)	`EMR_AutoScaling_DefaultRole`	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 with a custom policy for instance groups (p. 625).	`AmazonElasticMapReduceforAuto` For more information, see Service role for automatic scaling in EMR (Auto Scaling role) (p. 408).
Service role for EMR Notebooks (p. 409)	`EMR_Notebooks_DefaultRole`	Provides permissions that an EMR notebook needs to access other AWS resources and perform actions. Required only if EMR Notebooks is used.	`AmazonElasticMapReduceEditors` For more information, see Service role for EMR Notebooks (p. 409). `S3FullAccessPolicy` is also attached by default. Following is the contents of this policy. `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": "s3:*", "Resource": "*" } ] }`

Function	Default role	Description	Default managed policy
Service-Linked Role (p. 412)	`AWSServiceRoleForEMRCleanup`	Amazon 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 clean up. If a cluster uses Spot Instances, the permissions policy attached to the Service role for Amazon EMR (EMR role) (p. 398) must allow the creation of a service-linked role. For more information, see Service-linked role permissions for Amazon EMR (p. 413).	`AmazonEMRCleanupPolicy`

Topics

- IAM service roles used by Amazon EMR (p. 397)
- Customize IAM roles (p. 418)
- Configure IAM roles for EMRFS requests to Amazon S3 (p. 420)
- Use resource-based policies for Amazon EMR access to AWS Glue Data Catalog (p. 424)
- Use IAM roles with applications that call AWS services directly (p. 424)
- Allow users and groups to create and modify roles (p. 425)

IAM service roles used by Amazon EMR

Amazon EMR uses IAM service roles to perform actions on your behalf when provisioning cluster resources, running applications, dynamically scaling resources, and creating and running EMR Notebooks. Amazon EMR uses the following roles when interacting with other AWS services. Each role has a unique function within Amazon EMR. The topics in this section describe the role function and provide the default roles and permissions policy for each role.

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. 424).

Topics

- Service role for Amazon EMR (EMR role) (p. 398)
- Service role for cluster EC2 instances (EC2 instance profile) (p. 403)
- Service role for automatic scaling in EMR (Auto Scaling role) (p. 408)
- Service role for EMR Notebooks (p. 409)
- Using the service-linked role for Amazon EMR (p. 412)

Service role for Amazon EMR (EMR role)

The EMR role defines the allowable actions for Amazon EMR when provisioning resources and performing service-level tasks that are not performed in the context of an EC2 instance running within a cluster. For example, the service role is used to provision EC2 instances when a cluster launches.

- The default role name is `EMR_DefaultRole`.
- The Amazon EMR scoped default managed policy attached to `EMR_DefaultRole` is `AmazonEMRServicePolicy_v2`. This v2 policy replaces the deprecated default managed policy, `AmazonElasticMapReduceRole`.

`AmazonEMRServicePolicy_v2` depends on scoped down access to resources that EMR provisions or uses. When you use this policy, you need to pass the user tag `for-use-with-amazon-emr-managed-policies = true` when provisioning the cluster. EMR will automatically propagate those tags. Additionally, you may need to manually add a user tag to specific types of resources, such as EC2 security groups that were not created by EMR. See Tagging resources to use managed policies (p. 429).

The following shows the contents of the current `AmazonEMRServicePolicy_v2` policy. You can also see the current content of the AmazonEMRServicePolicy_v2 managed policy on the IAM console.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "CreateInTaggedNetwork",
   "Effect": "Allow",
   "Action": [
    "ec2:CreateNetworkInterface",
    "ec2:RunInstances",
    "ec2:CreateFleet",
    "ec2:CreateLaunchTemplate",
    "ec2:CreateLaunchTemplateVersion"
   ],
   "Resource": [
    "arn:aws:ec2:*:*:subnet/*",
    "arn:aws:ec2:*:*:security-group/*"
   ],
   "Condition": {
    "StringEquals": {
     "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
    }
   }
  },
  {
   "Sid": "CreateWithEMRTaggedLaunchTemplate",
   "Effect": "Allow",
   "Action": [
    "ec2:CreateFleet",
    "ec2:RunInstances",
    "ec2:CreateLaunchTemplateVersion"
   ],
   "Resource": "arn:aws:ec2:*:*:launch-template/*",
   "Condition": {
    "StringEquals": {
     "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
    }
   }
  },
  {
   "Sid": "CreateEMRTaggedLaunchTemplate",
   "Effect": "Allow",
   "Action": "ec2:CreateLaunchTemplate",
```

```
  "Resource": "arn:aws:ec2:*:*:launch-template/*",
  "Condition": {
   "StringEquals": {
    "aws:RequestTag/for-use-with-amazon-emr-managed-policies": "true"
   }
  }
 },
 {
  "Sid": "CreateEMRTaggedInstancesAndVolumes",
  "Effect": "Allow",
  "Action": [
   "ec2:RunInstances",
   "ec2:CreateFleet"
  ],
  "Resource": [
   "arn:aws:ec2:*:*:instance/*",
   "arn:aws:ec2:*:*:volume/*"
  ],
  "Condition": {
   "StringEquals": {
    "aws:RequestTag/for-use-with-amazon-emr-managed-policies": "true"
   }
  }
 },
 {
  "Sid": "ResourcesToLaunchEC2",
  "Effect": "Allow",
  "Action": [
   "ec2:RunInstances",
   "ec2:CreateFleet",
   "ec2:CreateLaunchTemplate",
   "ec2:CreateLaunchTemplateVersion"
  ],
  "Resource": [
   "arn:aws:ec2:*:*:network-interface/*",
   "arn:aws:ec2:*::image/ami-*",
   "arn:aws:ec2:*:*:key-pair/*",
   "arn:aws:ec2:*:*:capacity-reservation/*",
   "arn:aws:ec2:*:*:placement-group/pg-*",
   "arn:aws:ec2:*:*:fleet/*",
   "arn:aws:ec2:*:*:dedicated-host/*",
   "arn:aws:resource-groups:*:*:group/*"
  ]
 },
 {
  "Sid": "ManageEMRTaggedResources",
  "Effect": "Allow",
  "Action": [
   "ec2:CreateLaunchTemplateVersion",
   "ec2:DeleteLaunchTemplate",
   "ec2:DeleteNetworkInterface",
   "ec2:ModifyInstanceAttribute",
   "ec2:TerminateInstances"
  ],
  "Resource": "*",
  "Condition": {
   "StringEquals": {
    "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
   }
  }
 },
 {
  "Sid": "ManageTagsOnEMRTaggedResources",
  "Effect": "Allow",
  "Action": [
   "ec2:CreateTags",
```

```
    "ec2:DeleteTags"
   ],
   "Resource": [
    "arn:aws:ec2:*:*:instance/*",
    "arn:aws:ec2:*:*:volume/*",
    "arn:aws:ec2:*:*:network-interface/*",
    "arn:aws:ec2:*:*:launch-template/*"
   ],
   "Condition": {
    "StringEquals": {
     "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
    }
   }
  },
  {
   "Sid": "CreateNetworkInterfaceNeededForPrivateSubnet",
   "Effect": "Allow",
   "Action": [
    "ec2:CreateNetworkInterface"
   ],
   "Resource": [
    "arn:aws:ec2:*:*:network-interface/*"
   ],
   "Condition": {
    "StringEquals": {
     "aws:RequestTag/for-use-with-amazon-emr-managed-policies": "true"
    }
   }
  },
  {
   "Sid": "TagOnCreateTaggedEMRResources",
   "Effect": "Allow",
   "Action": [
    "ec2:CreateTags"
   ],
   "Resource": [
    "arn:aws:ec2:*:*:network-interface/*",
    "arn:aws:ec2:*:*:instance/*",
    "arn:aws:ec2:*:*:volume/*",
    "arn:aws:ec2:*:*:launch-template/*"
   ],
   "Condition": {
    "StringEquals": {
     "ec2:CreateAction": [
      "RunInstances",
      "CreateFleet",
      "CreateLaunchTemplate",
      "CreateNetworkInterface"
     ]
    }
   }
  },
  {
   "Sid": "TagPlacementGroups",
   "Effect": "Allow",
   "Action": [
    "ec2:CreateTags",
    "ec2:DeleteTags"
   ],
   "Resource": [
    "arn:aws:ec2:*:*:placement-group/pg-*"
   ]
  },
  {
   "Sid": "ListActionsForEC2Resources",
   "Effect": "Allow",
```

```
  "Action": [
   "ec2:DescribeAccountAttributes",
   "ec2:DescribeCapacityReservations",
   "ec2:DescribeDhcpOptions",
   "ec2:DescribeImages",
   "ec2:DescribeInstances",
   "ec2:DescribeLaunchTemplates",
   "ec2:DescribeNetworkAcls",
   "ec2:DescribeNetworkInterfaces",
   "ec2:DescribePlacementGroups",
   "ec2:DescribeRouteTables",
   "ec2:DescribeSecurityGroups",
   "ec2:DescribeSubnets",
   "ec2:DescribeVolumes",
   "ec2:DescribeVolumeStatus",
   "ec2:DescribeVpcAttribute",
   "ec2:DescribeVpcEndpoints",
   "ec2:DescribeVpcs"
  ],
  "Resource": "*"
 },
 {
  "Sid": "CreateDefaultSecurityGroupWithEMRTags",
  "Effect": "Allow",
  "Action": [
   "ec2:CreateSecurityGroup"
  ],
  "Resource": [
   "arn:aws:ec2:*:*:security-group/*"
  ],
  "Condition": {
   "StringEquals": {
    "aws:RequestTag/for-use-with-amazon-emr-managed-policies": "true"
   }
  }
 },
 {
  "Sid": "CreateDefaultSecurityGroupInVPCWithEMRTags",
  "Effect": "Allow",
  "Action": [
   "ec2:CreateSecurityGroup"
  ],
  "Resource": [
   "arn:aws:ec2:*:*:vpc/*"
  ],
  "Condition": {
   "StringEquals": {
    "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
   }
  }
 },
 {
  "Sid": "TagOnCreateDefaultSecurityGroupWithEMRTags",
  "Effect": "Allow",
  "Action": [
   "ec2:CreateTags"
  ],
  "Resource": "arn:aws:ec2:*:*:security-group/*",
  "Condition": {
   "StringEquals": {
    "aws:RequestTag/for-use-with-amazon-emr-managed-policies": "true",
    "ec2:CreateAction": "CreateSecurityGroup"
   }
  }
 },
 {
```

```
  "Sid": "ManageSecurityGroups",
  "Effect": "Allow",
  "Action": [
   "ec2:AuthorizeSecurityGroupEgress",
   "ec2:AuthorizeSecurityGroupIngress",
   "ec2:RevokeSecurityGroupEgress",
   "ec2:RevokeSecurityGroupIngress"
  ],
  "Resource": "*",
  "Condition": {
   "StringEquals": {
    "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
   }
  }
 },
 {
  "Sid": "CreateEMRPlacementGroups",
  "Effect": "Allow",
  "Action": [
   "ec2:CreatePlacementGroup"
  ],
  "Resource": "arn:aws:ec2:*:*:placement-group/pg-*"
 },
 {
  "Sid": "DeletePlacementGroups",
  "Effect": "Allow",
  "Action": [
   "ec2:DeletePlacementGroup"
  ],
  "Resource": "*"
 },
 {
  "Sid": "AutoScaling",
  "Effect": "Allow",
  "Action": [
   "application-autoscaling:DeleteScalingPolicy",
   "application-autoscaling:DeregisterScalableTarget",
   "application-autoscaling:DescribeScalableTargets",
   "application-autoscaling:DescribeScalingPolicies",
   "application-autoscaling:PutScalingPolicy",
   "application-autoscaling:RegisterScalableTarget"
  ],
  "Resource": "*"
 },
 {
  "Sid": "ResourceGroupsForCapacityReservations",
  "Effect": "Allow",
  "Action": [
   "resource-groups:ListGroupResources"
  ],
  "Resource": "*"
 },
 {
  "Sid": "AutoScalingCloudWatch",
  "Effect": "Allow",
  "Action": [
   "cloudwatch:PutMetricAlarm",
   "cloudwatch:DeleteAlarms",
   "cloudwatch:DescribeAlarms"
  ],
  "Resource": "arn:aws:cloudwatch:*:*:alarm:*_EMR_Auto_Scaling"
 },
 {
  "Sid": "PassRoleForAutoScaling",
  "Effect": "Allow",
  "Action": "iam:PassRole",
```

```
    "Resource": "arn:aws:iam::*:role/EMR_AutoScaling_DefaultRole",
    "Condition": {
     "StringLike": {
      "iam:PassedToService": "application-autoscaling.amazonaws.com*"
     }
    }
   },
   {
    "Sid": "PassRoleForEC2",
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "arn:aws:iam::*:role/EMR_EC2_DefaultRole",
    "Condition": {
     "StringLike": {
      "iam:PassedToService": "ec2.amazonaws.com*"
     }
    }
   }
  ]
}
```

Your service role should use the following trust policy.

> **Important**
> The following trust policy includes the `aws:SourceArn` and `aws:SourceAccount` global condition keys, which limit the permissions that you give Amazon EMR to particular resources in your account. Using them can protect you against the confused deputy problem.

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "elasticmapreduce.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "<account-id>"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:elasticmapreduce:<region>:<account-id>:*"
                }
            }
        }
    ]
}
```

Service role for cluster EC2 instances (EC2 instance profile)

The service role for cluster EC2 instances (also called the EC2 instance profile for Amazon EMR) is a special type of service role that is assigned to every EC2 instance in an Amazon EMR cluster when the instance launches. Application processes that run on top of the Hadoop ecosystem assume this role for permissions to interact with other AWS services.

For more information about service roles for EC2 instances, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

> **Important**
> The default service role for cluster EC2 instances and its associated AWS default managed policy, `AmazonElasticMapReduceforEC2Role` are on the path to deprecation, with no

replacement AWS managed policies provided. You'll need to create and specify an instance profile to replace the deprecated role and default policy.

Default role and managed policy

- The default role name is `EMR_EC2_DefaultRole`.
- The `EMR_EC2_DefaultRole` default managed policy, `AmazonElasticMapReduceforEC2Role`, is on the path to deprecation and will not be replaced with another default managed policy. Instead of using a default managed policy for the EC2 instance profile, apply resource-based policies to S3 buckets and other resources that Amazon EMR needs, or use your own customer managed policy with an IAM role as an instance profile. For more information, see Creating a service role for cluster EC2 instances with least-privilege permissions (p. 405).

The following shows the contents of version 3 of `AmazonElasticMapReduceforEC2Role`.

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

Your service role should use the following trust policy.

> **Important**
> The following trust policy includes the `aws:SourceArn` and `aws:SourceAccount` global condition keys, which limit the permissions that you give Amazon EMR to particular resources in your account. Using them can protect you against the confused deputy problem.

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "<account-id>"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:elasticmapreduce:<region>:<account-id>:*"
                }
            }
        }
    ]
}
```

Creating a service role for cluster EC2 instances with least-privilege permissions

As a best practice, we strongly recommend that you create a service role for cluster EC2 instances and permissions policy that has the minimum permissions to other AWS services required by your application.

The default managed policy, `AmazonElasticMapReduceforEC2Role`, provides permissions that make it easy to launch an initial cluster. However, `AmazonElasticMapReduceforEC2Role` is on the path to deprecation and Amazon EMR will not provide a replacement AWS managed default policy for the deprecated role. To launch an initial cluster, you need to provide a customer managed resource-based or ID-based policy.

The following policy statements provide examples of the permissions required for different features of Amazon EMR. We recommend that you use these permissions to create a permissions policy that restricts access to only those features and resources that your cluster requires. All example policy statements use the *`us-west-2`* Region and the fictional AWS account ID *`123456789012`*. Replace these as appropriate for your cluster.

For more information about creating and specifying custom roles, see Customize IAM roles (p. 418).

> **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.

Reading and writing data to Amazon S3 using EMRFS

When an application running on an Amazon EMR cluster references data using the `s3://mydata` format, Amazon EMR uses the EC2 instance profile to make the request. Clusters typically read and write data to Amazon S3 in this way, and Amazon EMR uses the permissions attached to the service role for cluster EC2 instances by default. For more information, see Configure IAM roles for EMRFS requests to Amazon S3 (p. 420).

Because IAM roles for EMRFS will fall back to the permissions attached to the service role for cluster EC2 instances, as a best practice, we recommend that you use IAM roles for EMRFS, and limit the EMRFS and Amazon S3 permissions attached to the service role for cluster EC2 instances.

The sample statement below demonstrates the permissions that EMRFS requires to make requests to Amazon S3.

- *`my-data-bucket-in-s3-for-emrfs-reads-and-writes`* specifies the bucket in Amazon S3 where the cluster reads and writes data and all sub-folders using `/*`. Add only those buckets and folders that your application requires.
- The policy statement that allows dynamodb actions is required only if EMRFS consistent view is enabled. *`EmrFSMetadata`* specifies the default folder for EMRFS consistent view.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:CreateBucket",
                "s3:DeleteObject",
                "s3:GetBucketVersioning",
                "s3:GetObject",
                "s3:GetObjectTagging",
                "s3:GetObjectVersion",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListBucketVersions",
                "s3:ListMultipartUploadParts",
                "s3:PutBucketVersioning",
                "s3:PutObject",
                "s3:PutObjectTagging"
            ],
            "Resource": [
                "arn:aws:s3:::my-data-bucket-in-s3-for-emrfs-reads-and-writes",
                "arn:aws:s3:::my-data-bucket-in-s3-for-emrfs-reads-and-writes/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:CreateTable",
                "dynamodb:BatchGetItem",
                "dynamodb:BatchWriteItem",
                "dynamodb:PutItem",
                "dynamodb:DescribeTable",
                "dynamodb:DeleteItem",
                "dynamodb:GetItem",
                "dynamodb:Scan",
```

```
                "dynamodb:Query",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteTable",
                "dynamodb:UpdateTable"
            ],
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/EmrFSMetadata"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData",
                "dynamodb:ListTables",
                "s3:ListBucket"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "sqs:GetQueueUrl",
                "sqs:ReceiveMessage",
                "sqs:DeleteQueue",
                "sqs:SendMessage",
                "sqs:CreateQueue"
            ],
            "Resource": "arn:aws:sqs:us-west-2:123456789012:EMRFS-Inconsistency-*"
        }
    ]
}
```

Archiving log files to Amazon S3

The following policy statement allows the Amazon EMR cluster to archive log files to the Amazon S3 location specified. In the example below, when the cluster was created, *s3://MyLoggingBucket/MyEMRClusterLogs* was specified using the **Log folder S3 location** in the console, using the `--log-uri` option from the AWS CLI, or using the `LogUri` parameter in the `RunJobFlow` command. For more information, see Archive log files to Amazon S3 (p. 343).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::MyLoggingBucket/MyEMRClusterLogs/*"
        }
    ]
}
```

Using the debugging tools

The following policy statement allows actions that are required if you enable the Amazon EMR debugging tool. Archiving log files to Amazon S3, and the associated permissions shown in the example above, are required for debugging. For more information, see Enable the debugging tool (p. 347).

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

```
                "sqs:GetQueueUrl",
                "sqs:SendMessage"
            ],
            "Resource": "arn:aws:sqs:us-west-2:123456789012:AWS-ElasticMapReduce-*"
        }
    ]
}
```

Using the AWS Glue Data Catalog

The following policy statement allows actions that are required if you use the AWS Glue Data Catalog as the metastore for applications. For more information, see Using the AWS Glue Data Catalog as the metastore for Spark SQL, Using the AWS Glue Data Catalog as the metastore for Hive, and Using Presto with the AWS Glue Data Catalog in the *Amazon EMR Release Guide*.

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

Service role for automatic scaling in EMR (Auto Scaling role)

The Auto Scaling role for EMR performs a similar function as the service role, but allows additional actions for dynamically scaling environments.

- The default role name is `EMR_AutoScaling_DefaultRole`.
- The default managed policy attached to `EMR_AutoScaling_DefaultRole` is `AmazonElasticMapReduceforAutoScalingRole`.

The contents of version 1 of `AmazonElasticMapReduceforAutoScalingRole` are shown below.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudwatch:DescribeAlarms",
                "elasticmapreduce:ListInstanceGroups",
                "elasticmapreduce:ModifyInstanceGroups"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Your service role should use the following trust policy.

> **Important**
> The following trust policy includes the `aws:SourceArn` and `aws:SourceAccount` global condition keys, which limit the permissions that you give Amazon EMR to particular resources in your account. Using them can protect you against the confused deputy problem.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                    "elasticmapreduce.amazonaws.com",
                    "application-autoscaling.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "<account-id>"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:elasticmapreduce:<region>:<account-id>:*"
                }
            }
        }
    ]
}
```

Service role for EMR Notebooks

Each EMR notebook needs permissions to access other AWS resources and perform actions. The IAM policies attached to this service role provide permissions for the notebook to interoperate with other AWS services. When you create a notebook using the AWS Management Console, you specify an *AWS service role*. You can use the default role, `EMR_Notebooks_DefaultRole`, or specify a role that you create. If a notebook has not been created before, you can choose to create the default role.

- The default role name is `EMR_Notebooks_DefaultRole`.
- The default managed policies attached to `EMR_Notebooks_DefaultRole` are `AmazonElasticMapReduceEditorsRole` and `S3FullAccessPolicy`.

Your service role should use the following trust policy.

Important

The following trust policy includes the `aws:SourceArn` and `aws:SourceAccount` global condition keys, which limit the permissions that you give Amazon EMR to particular resources in your account. Using them can protect you against the confused deputy problem.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "elasticmapreduce.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "<account-id>"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:elasticmapreduce:<region>:<account-id>:*"
                }
            }
        }
    ]
}
```

The contents of version 1 of `AmazonElasticMapReduceEditorsRole` are 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",
                "ec2:DescribeVpcs",
                "elasticmapreduce:ListInstances",
                "elasticmapreduce:DescribeCluster",
                "elasticmapreduce:ListSteps"
            ],
            "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"
                    ]
                }
            }
        }
    ]
}
```

Following is the contents of the `S3FullAccessPolicy`. The `S3FullAccessPolicy` allows your service role for EMR Notebooks to perform all Amazon S3 actions on objects in your AWS account. When you create a custom service role for EMR Notebooks, you must give your service role Amazon S3 permissions.

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

You can scope down read and write access for your service role to the Amazon S3 location where you want to save your notebook files. Use the following minimum set of Amazon S3 permissions.

```
"s3:PutObject",
"s3:GetObject",
"s3:GetEncryptionConfiguration",
"s3:ListBucket",
"s3:DeleteObject"
```

If your Amazon S3 bucket is encrypted, you must include the following permissions for AWS Key Management Service.

```
"kms:Decrypt",
"kms:GenerateDataKey",
"kms:ReEncryptFrom",
"kms:ReEncryptTo",
"kms:DescribeKey"
```

When you link Git repositories to your notebook and need to create a secret for the repository, you must add the `secretsmanager:GetSecretValue` permission in the IAM policy attached to the service role for EMR notebooks. An example policy is demonstrated below:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "*"
        }
    ]
}
```

EMR Notebooks service role permissions

This table lists the actions that EMR Notebooks takes using the service role, along with the permissions needed for each action.

Action	Permissions

Establish a secure network channel between a notebook and an Amazon EMR cluster, and perform necessary cleanup actions.	`"ec2:CreateNetworkInterface",`
	`"ec2:CreateNetworkInterfacePermission",`
	`"ec2:DeleteNetworkInterface",`
	`"ec2:DeleteNetworkInterfacePermission",`
	`"ec2:DescribeNetworkInterfaces",`
	`"ec2:ModifyNetworkInterfaceAttribute",`
	`"ec2:AuthorizeSecurityGroupEgress",`
	`"ec2:AuthorizeSecurityGroupIngress",`
	`"ec2:CreateSecurityGroup",`
	`"ec2:DescribeSecurityGroups",`
	`"ec2:RevokeSecurityGroupEgress",`
	`"ec2:DescribeTags",`
	`"ec2:DescribeInstances",`
	`"ec2:DescribeSubnets",`
	`"ec2:DescribeVpcs",`
	`"elasticmapreduce:ListInstances",`
	`"elasticmapreduce:DescribeCluster",`
	`"elasticmapreduce:ListSteps"`

Use Git credentials stored in AWS Secrets Manager to link Git repositories to a notebook.	`"secretsmanager:GetSecretValue"`
Apply AWS tags to the network interface and default security groups that EMR Notebooks creates while setting up the secure network channel. For more information, see Tagging AWS resources.	`"ec2:CreateTags"`

Access or upload notebook files and metadata to Amazon S3.	`"s3:PutObject",`
	`"s3:GetObject",`
	`"s3:GetEncryptionConfiguration",`
	`"s3:ListBucket",`
	`"s3:DeleteObject"`
	
	The following permissions are only required if you use an encrypted Amazon S3 bucket.
	
	`"kms:Decrypt",`
	`"kms:GenerateDataKey",`
	`"kms:ReEncryptFrom",`
	`"kms:ReEncryptTo",`
	`"kms:DescribeKey"`

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 service roles for Amazon EMR permissions to AWS services and resources (p. 394).

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: `ModifyInstanceAttribute` 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. 413).

> **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 right of the **Role description**, choose **Edit**.
4. Enter a new description in the box and choose **Save changes**.

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. 607).

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)	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

Region name	Region identity	Support in Amazon EMR
Canada (Central)	ca-central-1	Yes
Europe (Frankfurt)	eu-central-1	Yes
Europe (Ireland)	eu-west-1	Yes
Europe (London)	eu-west-2	Yes
Europe (Paris)	eu-west-3	Yes
South America (São Paulo)	sa-east-1	Yes

Customize IAM roles

You may want to customize the IAM service roles and permissions to limit privileges according to your security requirements. 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. 425).

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. 418).

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. 425).

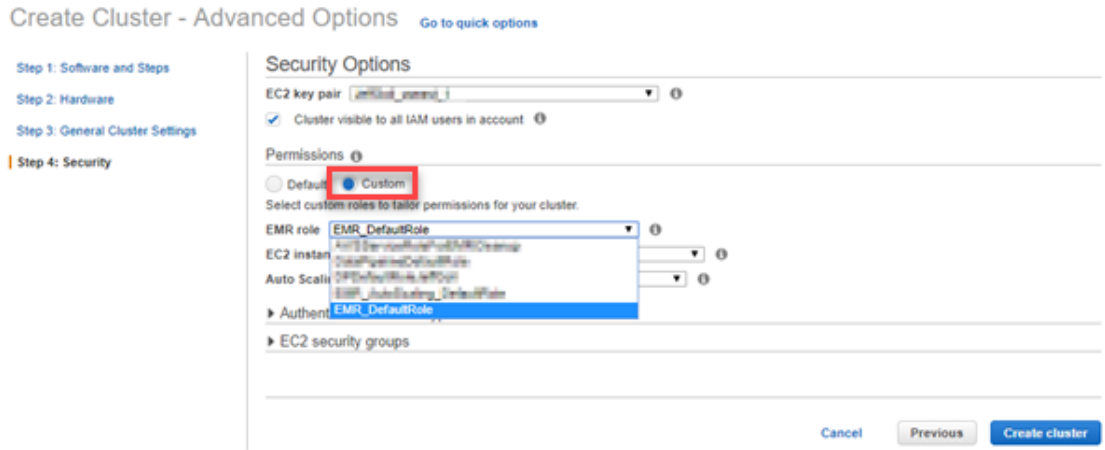
Use the console to specify custom roles

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 IAM service roles used by Amazon EMR (p. 397).

To specify custom IAM roles using the console

1. Open the Amazon EMR console at https://console.aws.amazon.com/emr.
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 for EMR and a service role for cluster EC2 instances explicitly 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.

The Auto Scaling role is specified using a separate option, `--auto-scaling-role`. For more information, see Using automatic scaling with a custom policy for instance groups (p. 625).

To specify custom IAM roles using the AWS CLI

- The following command specifies the custom service role, *`MyCustomServiceRoleForEMR`*, and a custom role for the EC2 instance profile, *`MyCustomServiceRoleForClusterEC2Instances`*, 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-5.36.0 \
--applications Name=Hive Name=Pig --service-role MyCustomServiceRoleForEMR \
--ec2-attributes InstanceProfile=MyCustomServiceRoleForClusterEC2Instances,\
KeyName=myKey --instance-type m5.xlarge --instance-count 3
```

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 `config` file for the AWS CLI.

The following example demonstrates the contents of a `config` file for the AWS CLI the specifies custom roles for Amazon EMR. With this configuration file, when the `--use-default-roles` option is specified, the cluster is created using the *`MyCustomServiceRoleForEMR`* and *`MyCustomServiceRoleForClusterEC2Instances`*. By default, the `config` file specifies the default `service_role` as `AmazonElasticMapReduceRole` and the default `instance_profile` as `EMR_EC2_DefaultRole`.

```
[default]
output = json
region = us-west-1
aws_access_key_id = myAccessKeyID
aws_secret_access_key = mySecretAccessKey
emr =
     service_role = MyCustomServiceRoleForEMR
     instance_profile = MyCustomServiceRoleForClusterEC2Instances
```

Configure IAM roles for EMRFS requests to Amazon S3

When an application running on a cluster references data using the `s3://mydata` format, Amazon EMR uses EMRFS to make the request. To interact with Amazon S3, EMRFS assumes the permissions policies that are attached to your Amazon EC2 instance profile (p. 403). The same Amazon EC2 instance profile is used regardless of the user or group running the application or the location of the data in Amazon S3.

If you have a cluster with multiple users who need different levels of access to data in Amazon S3 through EMRFS, you can set up a security configuration with IAM roles for EMRFS. EMRFS can assume a different service role for cluster EC2 instances based on the user or group making the request, or based on the location of data in Amazon S3. Each IAM role for EMRFS can have different permissions for data access in Amazon S3. For more information about the service role for cluster EC2 instances, see Service role for cluster EC2 instances (EC2 instance profile) (p. 403).

Using custom IAM roles for EMRFS is supported in Amazon EMR versions 5.10.0 and later. If you use an earlier 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.

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. These identifiers 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, if the request matches the basis for access, EMRFS has cluster EC2 instances assume the corresponding IAM role for the request. The IAM permissions attached to that role apply instead of the IAM permissions attached to the service role for cluster EC2 instances.

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.

> **Important**
> IAM roles for EMRFS provide application-level isolation between users of the application. It does not provide host level isolation between users on the host. Any user with access to the cluster can bypass the isolation to assume any of the roles.

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 service role for cluster EC2 instances. For this reason, we recommend that the policies attached to this role limit permissions

to Amazon S3. For more information, see Service role for cluster EC2 instances (EC2 instance profile) (p. 403).

Configure roles

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, and then edit this policy according to your requirements. The default role name is `EMR_EC2_DefaultRole`, and the default managed policy to edit is `AmazonElasticMapReduceforEC2Role`. For more information, see Service role for cluster EC2 instances (EC2 instance profile) (p. 403).

Updating trust policies to assume role permissions

Each role that EMRFS uses must have a trust policy that allows the cluster's EMR role for EC2 to assume it. Similarly, the cluster's EMR role for EC2 must have a trust policy that allows EMRFS roles to assume it.

The following example trust policy is attached to roles for EMRFS. The statement allows the default EMR role for EC2 to assume the role. For example, if you have two fictitious EMRFS roles, `EMRFSRole_First` and `EMRFSRole_Second`, this policy statement is added to the trust policies for each of them.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::AWSAcctID:role/EMR_EC2_DefaultRole"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

In addition, the following example trust policy statement is added to the `EMR_EC2_DefaultRole` to allow the two fictitious EMRFS roles to assume it.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "AWS": ["arn:aws:iam::AWSAcctID:role/EMRFSRole_First",
 "arn:aws:iam::AWSAcctID:role/EMRFSRole_Second"]
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

To update the trust policy of an IAM role

Open the IAM console at https://console.aws.amazon.com/iam/.

1. Choose **Roles**, enter the name of the role in **Search**, and then select its **Role name**.
2. Choose **Trust relationships**, **Edit trust relationship**.
3. Add a trust statement according to the **Policy document** according to the guidelines above, and then choose **Update trust policy**.

Specifying a role as a key user

If a role allows access to a location in Amazon S3 that is encrypted using an AWS KMS key, make sure that the role is specified as a key user. This gives the role permission to use the KMS key. For more information, see Key policies in AWS KMS in the *AWS Key Management Service Developer Guide*.

Set up a security configuration with IAM roles for EMRFS

> **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. 418) and Specify custom IAM roles when you create a cluster (p. 418).

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. 422) 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. 355).
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. 373).

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. 355).

   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 `arn:aws:iam::`*`account-id`*`:role/`*`role-name`*. This is the IAM role that Amazon EMR assumes if the EMRFS request to Amazon S3 matches any of the `Identifiers` specified.
`"IdentifierType":`	Can be one of the following: • `"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 `s3://mybucket/` matches `s3://mybucket/mydir` and `s3://mybucket/yetanotherdir`. • `"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.

Parameter	Description
`"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-5.36.0 --ec2-attributes
 InstanceProfile=EC2_Role_EMR_Restrict_S3,KeyName=MyKey \
--instance-type m5.xlarge --instance-count 3 \
--security-configuration EMRFS_Roles_Security_Configuration
```

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, for a resource-based policy attached to a catalog, you can specify the role ARN for the default service role for cluster EC2 instances, *`EMR_EC2_DefaultRole`* as the `Principal`, using the format shown in the following example:

```
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 EC2 instance profile to obtain temporary security credentials when calling AWS services.

The versions of Hadoop available with Amazon EMR release 2.3.0 and later have 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 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 following example 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.

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.

For information about assigning permissions to an IAM user, see Changing permissions for an IAM user in the *IAM User Guide*.

Amazon EMR identity-based policy examples

By default, IAM users and roles don't have permission to create or modify Amazon EMR resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform specific API operations on the specified resources they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see Creating policies on the JSON tab in the *IAM User Guide*.

Topics

- Policy best practices for Amazon EMR (p. 426)
- Allow users to view their own permissions (p. 426)
- Amazon EMR managed policies (p. 427)
- IAM policies for tag-based access to clusters and EMR notebooks (p. 437)
- Denying the ModifyInstanceGroup action (p. 445)

Policy best practices for Amazon EMR

Identity-based policies are very powerful. They determine whether someone can create, access, or delete Amazon EMR resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get Started Using AWS Managed Policies** – To start using Amazon EMR quickly, use AWS managed policies to give your employees the permissions they need. These policies are already available in your account and are maintained and updated by AWS. For more information, see Get started using permissions with AWS managed policies in the *IAM User Guide* and Amazon EMR managed policies (p. 427).
- **Grant Least Privilege** – When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. For more information, see Grant least privilege in the *IAM User Guide*.
- **Enable MFA for Sensitive Operations** – For extra security, require IAM users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.
- **Use Policy Conditions for Extra Security** – To the extent that it's practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to specify a range of allowable IP addresses that a request must come from. You can also write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
```

```
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"ViewOwnUserInfo",
         "Effect":"Allow",
         "Action":[
            "iam:GetUser",
            "iam:GetUserPolicy",
            "iam:ListAttachedUserPolicies",
            "iam:ListGroupsForUser",
            "iam:ListUserPolicies"
         ],
         "Resource":[
            "arn:aws:iam:::user/${aws:username}"
         ]
      },
      {
         "Sid":"NavigateInConsole",
         "Effect":"Allow",
         "Action":[
            "iam:GetGroupPolicy",
            "iam:GetPolicy",
            "iam:GetPolicyVersion",
            "iam:ListAttachedGroupPolicies",
            "iam:ListGroupPolicies",
            "iam:ListGroups",
            "iam:ListPolicies",
            "iam:ListPolicyVersions",
            "iam:ListUsers"
         ],
         "Resource":""
      }
   ]
}
```

Amazon EMR managed policies

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.

Amazon EMR will be deprecating existing managed policies (v1 policies) in favor of new managed policies (v2 policies). The new managed policies have been scoped-down to align with AWS best practices. After the existing v1 managed policies are deprecated, you will not be able to attach these policies to any new IAM roles or users. Existing roles and users that use deprecated policies can continue to use them. The v2 managed policies restrict access using tags. They allow only specified Amazon EMR actions and require cluster resources that are tagged with an EMR-specific key. We recommend that you carefully review the documentation before using the new v2 policies.

The v1 policies will be marked deprecated with a warning icon next to them in the **Policies** list in the IAM console. The deprecated policies will have the following characteristics:

- They will continue to work for all currently attached users, groups, and roles. Nothing breaks.
- They cannot be attached to new users, groups, or roles. If you detach one of the policies from a current entity, you cannot reattach it.
- After you detach a v1 policy from all current entities, the policy will no longer be visible and can no longer be used.

The following table summarizes the changes between current policies (v1) and v2 policies.

EMR managed policy changes

Policy type	Policy names	Policy purpose	Changes to v2 policy
IAM managed policy for full EMR access by attached user, role, or group	V1 policy (to be deprecated): **AmazonElasticMapReduceFullAccess** (p. 434) V2 (scoped) policy name: **AmazonEMRFullAccessPolicy_v2** (p. 431)	Allows users full permissions for EMR actions. Includes iam:PassRole permissions for resources.	Policy adds a prerequisite that users must add user tags to resources before they can use this policy. See Tagging resources to use managed policies (p. 429). iam:PassRole action requires iam:PassedToService condition set to specified service. Access to Amazon EC2, Amazon S3, and other services is not allowed by default. See IAM Managed Policy for Full Access (v2 Managed Default Policy) (p. 431).
IAM managed policy for read-only access by attached user, role, or group	V1 policy (to be deprecated): **AmazonElasticMapReduceReadOnlyAccess** (p. 435) V2 (scoped) policy name: **AmazonEMRReadOnlyAccessPolicy_v2** (p. 434)	Allows users read-only permissions for Amazon EMR actions.	Permissions allow only specified elasticmapreduce read-only actions. Access to Amazon S3 is access not allowed by default. See IAM Managed Policy for Read-Only Access (v2 Managed Default Policy) (p. 434).
Default EMR service role and attached managed policy	Role name: **EMR_DefaultRole** V1 policy (to be deprecated): **AmazonElasticMapReduceRole** (EMR Service Role) V2 (scoped-down) policy name: **AmazonEMRServicePolicy_v2** (p. 398)	Allows Amazon EMR to call other AWS services on your behalf when provisioning resources and performing service-level actions. This role is required for all clusters.	The v2 service role and v2 default policy replace the deprecated role and policy. The policy adds a prerequisite that users must add user tags to resources before they can use this policy. See Tagging resources to use managed policies (p. 429). See Service role for Amazon EMR (EMR role) (p. 398).
Service role for cluster EC2 instances (EC2 instance profile)	V1 policy (to be deprecated): **EMR_EC2_DefaultRole** (instance profile)	Allows applications running on an EMR cluster to access other AWS resources, such	Both default role and default policy are on the path to deprecation. There is

Policy type	Policy names	Policy purpose	Changes to v2 policy
	Deprecated policy name: **AmazonElasticMapReduceforEC2Role**	for EC2 Spark jobs that process data from Amazon S3, the policy needs to allow access to such resources. as Amazon S3. For example, if you run	no replacement AWS default managed role or policy. You need to provide a resource-based or identity-based policy. This means that, by default, applications running on an EMR cluster do not have access to Amazon S3 or other resource unless you manually add these to the policy. See Default role and managed policy (p. 404).
Other EC2 service role policies	Current policy names: **AmazonElasticMapReduceforAutoScalingRole, AmazonElasticMapReduceEditorRole, AmazonEMRCleanupPolicy**	Provides permissions for AWS resources and perform actions if using auto scaling, notebooks, or to clean up EC2 resources.	No changes for v2.

Securing iam:PassRole

The Amazon EMR full permissions default managed policy (`AmazonEMRFullAccessPolicy_v2`) and Amazon EMR service policy (`AmazonEMRServicePolicy_v2`) are available to replace the soon-to-be-deprecated policy. The v2 policies incorporate new iam:PassRole security configurations, including the following:

- `iam:PassRole` permissions only for specific default Amazon EMR roles.
- `iam:PassedToService` conditions that allow you to use the policy with only specified AWS services, such as `elasticmapreduce.amazonaws.com` and `ec2.amazonaws.com`.

You can view the JSON version of the AmazonEMRFullAccessPolicy_v2 and AmazonEMRServicePolicy_v2 policies in the IAM console.

We recommend that you create new clusters using v2 managed policies.

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*.

Tagging resources to use managed policies

AmazonEMRServicePolicy_v2 and AmazonEMRFullAccessPolicy_v2 depend on scoped-down access to resources that Amazon EMR provisions or uses. The scope down is achieved by restricting access to only those resources that have a predefined user tag associated with them. When you use either of these two policies, you need to pass the predefined user tag `for-use-with-amazon-emr-managed-`

`policies = true` when provisioning the cluster. Amazon EMR will then automatically propagate that tag. Additionally, you need to manually add a user tag to resources listed in the following section. If you are using the Amazon EMR console, see Considerations for using the Amazon EMR console to launch clusters with v2 managed policies (p. 431).

To use managed policies, you pass the user tag `for-use-with-amazon-emr-managed-policies = true` when provisioning a cluster using the CLI, SDK, or another method.

When you pass the tag, Amazon EMR propagates the tag to private subnet ENI, EC2 instance, and EBS volumes that it creates. Amazon EMR also automatically tags security groups that it creates. However, if you want Amazon EMR to launch with a certain security group, you must tag it. For resources that are not created by Amazon EMR, you need to manually add tags to those resources. For example, you must tag EC2 subnets, EC2 security groups (if not created by Amazon EMR), and VPCs (if you want Amazon EMR to create security groups). To launch clusters with v2 managed policies in VPCs, you must tag those VPCs with the predefined user tag. See, Considerations for using the Amazon EMR console to launch clusters with v2 managed policies (p. 431).

Propagated user-specified tagging

Amazon EMR tags resources that it creates using the Amazon EMR tags that you specify when creating a cluster. Amazon EMR applies tags to the resources it creates during the lifetime of the cluster.

Amazon EMR propagates user tags for the following resources:

- Private Subnet ENI (service access elastic network interfaces)
- EC2 Instances
- EBS Volumes
- EC2 Launch Template

Automatically-tagged security groups

Amazon EMR tags EC2 security groups that it creates with the tag that is required for v2 managed policies for Amazon EMR, `for-use-with-amazon-emr-managed-policies`, regardless of which tags you specify in the create cluster command. For a security group that was created before the introduction of v2 managed policies, Amazon EMR does not automatically tag the security group. If you want to use v2 managed policies with the default security groups that already exist in the account, you need to tag the security groups manually with `for-use-with-amazon-emr-managed-policies = true`.

Manually-tagged cluster resources

You must manually tag some cluster resources so that they can be accessed by Amazon EMR default roles.

- You must manually tag EC2 security groups and EC2 subnets with the Amazon EMR managed policy tag `for-use-with-amazon-emr-managed-policies`.
- You must manually tag a VPC if you want Amazon EMR to create default security groups. EMR will try to create a security group with the specific tag if the default security group doesn't already exist.

Amazon EMR automatically tags the following resources:

- EMR-created EC2 Security Groups

You must manually tag the following resources:

- EC2 Subnet
- EC2 Security Groups

Optionally, you can manually tag the following resources:

- VPC - only when you want Amazon EMR to create security groups

Considerations for using the Amazon EMR console to launch clusters with v2 managed policies

You can provision clusters with v2 managed policies using the Amazon EMR console. Here are some considerations when you use the console to launch Amazon EMR clusters.

- You do not need to pass the predefined tag. Amazon EMR automatically adds the tag and propagates it to the appropriate components.
- For components that need to be manually tagged, the Amazon EMR console tries to automatically tag them if you have the required permissions to tag resources. If you don't have the permissions to tag resources, ask your administrator to tag those resources.
- You cannot launch clusters with v2 managed policies unless all the prerequisites are met.
- The console shows you which resources (VPC/Subnets) need to be tagged.

IAM managed policy for full access (v2 managed default policy)

The v2 scoped EMR default managed policies grant specific access privileges to users. They require a predefined Amazon EMR resource tag and `iam:PassRole` condition keys to resources used by Amazon EMR, such as the `Subnet` and `SecurityGroup` you use to launch your cluster.

To grant required actions scoped for Amazon EMR, attach the `AmazonEMRFullAccessPolicy_v2` managed policy. This updated default managed policy replaces the `AmazonElasticMapReduceFullAccess` (p. 434) managed policy.

`AmazonEMRFullAccessPolicy_v2` depends on scoped-down access to resources that Amazon EMR provisions or uses. When you use this policy, you need to pass the user tag `for-use-with-amazon-emr-managed-policies = true` when provisioning the cluster. Amazon EMR will automatically propagate the tag. Additionally, you may need to manually add a user tag to specific types of resources, such as EC2 security groups that were not created by Amazon EMR. For more information, see Tagging resources to use managed policies (p. 429).

The `AmazonEMRFullAccessPolicy_v2` policy secures resources by doing the following:

- Requires resources to be tagged with the pre-defined Amazon EMR managed policies tag `for-use-with-amazon-emr-managed-policies` for cluster creation and Amazon EMR access.
- Restricts the `iam:PassRole` action to specific default roles and `iam:PassedToService` access to specific services.
- No longer provides access to Amazon EC2, Amazon S3, and other services by default.

Following are the contents of this policy.

> **Note**
> You can also use the console link `AmazonEMRFullAccessPolicy_v2` to view the policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RunJobFlowExplicitlyWithEMRManagedTag",
            "Effect": "Allow",
            "Action": [
```

```
                "elasticmapreduce:RunJobFlow"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/for-use-with-amazon-emr-managed-policies": "true"
                }
            }
        },
        {
            "Sid": "ElasticMapReduceActions",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:AddInstanceFleet",
                "elasticmapreduce:AddInstanceGroups",
                "elasticmapreduce:AddJobFlowSteps",
                "elasticmapreduce:AddTags",
                "elasticmapreduce:CancelSteps",
                "elasticmapreduce:CreateEditor",
                "elasticmapreduce:CreateSecurityConfiguration",
                "elasticmapreduce:DeleteEditor",
                "elasticmapreduce:DeleteSecurityConfiguration",
                "elasticmapreduce:DescribeCluster",
                "elasticmapreduce:DescribeEditor",
                "elasticmapreduce:DescribeJobFlows",
                "elasticmapreduce:DescribeSecurityConfiguration",
                "elasticmapreduce:DescribeStep",
                "elasticmapreduce:DescribeReleaseLabel",
                "elasticmapreduce:GetBlockPublicAccessConfiguration",
                "elasticmapreduce:GetManagedScalingPolicy",
                "elasticmapreduce:GetAutoTerminationPolicy",
                "elasticmapreduce:ListBootstrapActions",
                "elasticmapreduce:ListClusters",
                "elasticmapreduce:ListEditors",
                "elasticmapreduce:ListInstanceFleets",
                "elasticmapreduce:ListInstanceGroups",
                "elasticmapreduce:ListInstances",
                "elasticmapreduce:ListSecurityConfigurations",
                "elasticmapreduce:ListSteps",
                "elasticmapreduce:ModifyCluster",
                "elasticmapreduce:ModifyInstanceFleet",
                "elasticmapreduce:ModifyInstanceGroups",
                "elasticmapreduce:OpenEditorInConsole",
                "elasticmapreduce:PutAutoScalingPolicy",
                "elasticmapreduce:PutBlockPublicAccessConfiguration",
                "elasticmapreduce:PutManagedScalingPolicy",
                "elasticmapreduce:RemoveAutoScalingPolicy",
                "elasticmapreduce:RemoveManagedScalingPolicy",
                "elasticmapreduce:RemoveTags",
                "elasticmapreduce:SetTerminationProtection",
                "elasticmapreduce:StartEditor",
                "elasticmapreduce:StopEditor",
                "elasticmapreduce:TerminateJobFlows",
                "elasticmapreduce:ViewEventsFromAllClustersInConsole"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ViewMetricsInEMRConsole",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricStatistics"
            ],
            "Resource": "*"
        },
        {
```

```
            "Sid": "PassRoleForElasticMapReduce",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::*:role/EMR_DefaultRole",
                "arn:aws:iam::*:role/EMR_DefaultRole_V2"
            ],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "elasticmapreduce.amazonaws.com*"
                }
            }
        },
        {
            "Sid": "PassRoleForEC2",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::*:role/EMR_EC2_DefaultRole",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "ec2.amazonaws.com*"
                }
            }
        },
        {
            "Sid": "PassRoleForAutoScaling",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::*:role/EMR_AutoScaling_DefaultRole",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "application-autoscaling.amazonaws.com*"
                }
            }
        },
        {
            "Sid": "ElasticMapReduceServiceLinkedRole",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
elasticmapreduce.amazonaws.com*/AWSServiceRoleForEMRCleanup*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": [
                        "elasticmapreduce.amazonaws.com",
                        "elasticmapreduce.amazonaws.com.cn"
                    ]
                }
            }
        },
        {
            "Sid": "ConsoleUIActions",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeImages",
                "ec2:DescribeKeyPairs",
                "ec2:DescribeNatGateways",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DescribeVpcEndpoints",
                "s3:ListAllMyBuckets",
                "iam:ListRoles"
```

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

IAM managed policy for full access (on path to deprecation)

The `AmazonElasticMapReduceFullAccess` managed policy grants all the required actions for Amazon EMR and other services. The `AmazonElasticMapReduceFullAccess` managed policy is on the path to deprecation, but not yet deprecated. When the policy is eventually deprecated, you will not abe able to attach it to a role. However, a role that already has an attached deprecated policy can still be attached to a cluster. `AmazonElasticMapReduceFullAccess` has been replaced with `AmazonEMRFullAccessPolicy_v2` (p. 431) as the Amazon EMR default managed policy.

The Amazon EMR full permissions default managed policy (`AmazonEMRFullAccessPolicy_v2`) and Amazon EMR service policy (`AmazonEMRServicePolicy_v2`) are available to replace the soon-to-be-deprecated policy. The v2 policies incorporate new iam:PassRole security configurations, including the following:

- `iam:PassRole` permissions only for specific default Amazon EMR roles.
- `iam:PassedToService` conditions that allow you to use the policy with only specified AWS services, such as `elasticmapreduce.amazonaws.com` and `ec2.amazonaws.com`.

You can view the JSON version of the AmazonEMRFullAccessPolicy_v2 and AmazonEMRServicePolicy_v2 policies in the IAM console.

We recommend that you create new clusters using v2 managed policies.

You can view the contents of this policy using the AWS Management Console link `AmazonElasticMapReduceFullAccess`.

> **Note**
> The `ec2:TerminateInstances` action in the policy enables the IAM user or role that assumes this policy to terminate any of the Amazon EC2 instances associated with the IAM account, even those that are not part of an Amazon EMR cluster.

IAM managed policy for read-only access (v2 managed default policy)

To grant read-only privileges to Amazon EMR, attach the **AmazonEMRReadOnlyAccessPolicy_v2** managed policy. This default managed policy replaces the `AmazonElasticMapReduceReadOnlyAccess` (p. 435) managed policy. The content of this policy statement is shown in the following snippet. Compared with the `AmazonElasticMapReduceReadOnlyAccess` policy, the `AmazonEMRReadOnlyAccessPolicy_v2` policy does not use wildcard characters for the `elasticmapreduce` element. Instead, the default v2 policy scopes the specific `elasticmapreduce` actions that are allowed.

> **Note**
> You can also use the AWS Management Console link `AmazonEMRReadOnlyAccessPolicy_v2` to view the policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ElasticMapReduceActions",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:DescribeCluster",
```

```
                "elasticmapreduce:DescribeEditor",
                "elasticmapreduce:DescribeJobFlows",
                "elasticmapreduce:DescribeSecurityConfiguration",
                "elasticmapreduce:DescribeStep",
                "elasticmapreduce:DescribeReleaseLabel",
                "elasticmapreduce:GetBlockPublicAccessConfiguration",
                "elasticmapreduce:GetManagedScalingPolicy",
                "elasticmapreduce:GetAutoTerminationPolicy",
                "elasticmapreduce:ListBootstrapActions",
                "elasticmapreduce:ListClusters",
                "elasticmapreduce:ListEditors",
                "elasticmapreduce:ListInstanceFleets",
                "elasticmapreduce:ListInstanceGroups",
                "elasticmapreduce:ListInstances",
                "elasticmapreduce:ListSecurityConfigurations",
                "elasticmapreduce:ListSteps",
                "elasticmapreduce:ViewEventsFromAllClustersInConsole"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ViewMetricsInEMRConsole",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricStatistics"
            ],
            "Resource": "*"
        }
    ]
}
```

IAM managed policy for read-only access (on path to deprecation)

The `AmazonElasticMapReduceReadOnlyAccess` managed policy is on the path to deprecation. You cannot attach this policy when launching new clusters. `AmazonElasticMapReduceReadOnlyAccess` has been replaced with `AmazonEMRReadOnlyAccessPolicy_v2` (p. 434) as the Amazon EMR default managed policy. The content of this policy statement is shown in the following snippet. 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**
> You can also use the AWS Management Console to view the 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": "*"
    }
  ]
}
```

AWS managed policies for Amazon EMR

To add permissions to users, groups, and roles, it is easier to use AWS managed policies than to write policies yourself. It takes time and expertise to create IAM customer managed policies that provide your team with only the permissions they need. To get started quickly, you can use our AWS managed policies. These policies cover common use cases and are available in your AWS account. For more information about AWS managed policies, see AWS managed policies in the *IAM User Guide*.

AWS services maintain and update AWS managed policies. You can't change the permissions in AWS managed policies. Services occasionally add additional permissions to an AWS managed policy to support new features. This type of update affects all identities (users, groups, and roles) where the policy is attached. Services are most likely to update an AWS managed policy when a new feature is launched or when new operations become available. Services do not remove permissions from an AWS managed policy, so policy updates won't break your existing permissions.

Additionally, AWS supports managed policies for job functions that span multiple services. For example, the **ReadOnlyAccess** AWS managed policy provides read-only access to all AWS services and resources. When a service launches a new feature, AWS adds read-only permissions for new operations and resources. For a list and descriptions of job function policies, see AWS managed policies for job functions in the *IAM User Guide*.

Amazon EMR updates to AWS managed policies

View details about updates to AWS managed policies for Amazon EMR since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the Amazon EMR Document history page.

Change	Description	Date
AmazonEMRFullAccessPolicy_v2 (p. 431) and **AmazonEMRReadOnlyAccessPolicy_v2** (p. 434) – Update to an existing policy	Added `elasticmapreduce:DescribeReleaseLabel` and `elasticmapreduce:GetAutoTerminationPolicy`.	April 21, 2022
AmazonEMRFullAccessPolicy_v2 (p. 431) – Update to an existing policy	Added `ec2:DescribeImages` for Using a custom AMI (p. 181).	February 15, 2022
Amazon EMR managed policies (p. 427)	Updated to clarify use of predefined user tags. Added section on using the AWS console to launch clsuters with v2 managed policies.	September 29, 2021
AmazonEMRFullAccessPolicy_v2 (p. 431) – Update to an existing policy	Changed the `PassRoleForAutoScaling` and `PassRoleForEC2` actions to use the `StringLike` condition operator to match `"iam:PassedToService":"application-autoscaling.amazonaws.com*"` and	May 20, 2021

Change	Description	Date
	`"iam:PassedToService":"ec2.amazonaws.com*"`, respectively.	
`AmazonEMRFullAccessPolicy_v2` (p. 473) – Update to an existing policy	Removed invalid action `s3:ListBuckets` and replaced with `s3:ListAllMyBuckets` action. Updated service-linked role (SLR) creation to be explicitly scoped-down to the only SLR that Amazon EMR has with explicit Service Principles. The SLRs that can be created are exactly the same as before this change.	March 23, 2021
`AmazonEMRFullAccessPolicy_v2` (p. 473) – New policy	Amazon EMR added new permissions to scope access to resources and to add a prerequisite that users must add predefined user tag to resources before they can use Amazon EMR managed policies. `iam:PassRole` action requires `iam:PassedToService` condition set to specified service. Access to Amazon EC2, Amazon S3, and other services is not allowed by default.	March 11, 2021
`AmazonEMRServicePolicy_v2` (p. 448) – New policy	Adds a prerequisite that users must add user tags to resources before they can use this policy.	March 11, 2021
`AmazonEMRReadOnlyAccessPolicy_v2` (p. 474) – New policy	Permissions allow only specified elasticmapreduce read-only actions. Access to Amazon S3 is access not allowed by default.	March 11, 2021
Amazon EMR started tracking changes	Amazon EMR started tracking changes for its AWS managed policies.	March 11, 2021

IAM policies for tag-based access to clusters and EMR notebooks

You can use conditions in your identity-based policy to control access to clusters and EMR notebooks based on tags.

For more information about adding tags to clusters, see Tagging EMR clusters. For more information about using condition keys, see Condition keys (p. 387).

The following examples demonstrate different scenarios and ways to use condition operators with Amazon EMR condition 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.

> **Important**
> Explicitly denying permission for tagging actions is an important consideration. This prevents users from tagging a resource and thereby granting themselves permissions that you did not intend to grant. If tagging actions for a resource are not denied, a user can modify tags and circumvent the intention of the tag-based policies. For an example of a policy that denies tagging actions, see Deny access to add and remove tags (p. 439).

Example identity-based policy statements for clusters

The following examples demonstrate identity-based permissions policies that are used to control the actions that are allowed with EMR clusters.

> **Important**
> The `ModifyInstanceGroup` action in Amazon EMR does not require that you specify a cluster ID. For that reason, denying this action based on cluster tags requires additional consideration. For more information, see Denying the ModifyInstanceGroup action (p. 445).

Topics

- Allow actions only on clusters with specific tag values (p. 438)
- Require cluster tagging when a cluster is created (p. 439)
- Deny access to add and remove tags (p. 439)
- Allow actions on clusters with a specific tag, regardless of tag value (p. 440)

Allow actions only on clusters with specific tag values

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

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"]
              }
            }
```

Require cluster tagging when a cluster is created

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 key specifies that the policy applies during tag creation, so the user must create a cluster with a tag that matches the specified value. For the `PassRole` resource, you need to provide the AWS account ID or alias, and the service role name. For more information about the IAM ARN format, see IAM ARNs in the *IAM User Guide*.

For more information specifying matching tag-key values, see `aws:RequestTag/tag-key` in the *IAM User Guide*.

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Sid": "RunJobFlowExplicitlyWithTag",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:RunJobFlow"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/department": "dev"
                }
            }
        },
        {
            "Sid": "PolicyPassroleXYZ",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::AccountId:role/Role-Name-With-Path"
            ]
        }
    ]
}
```

Deny access to add and remove tags

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. This policy

prevents a user from adding or removing tags on EMR clusters with a *department* tag whose value is not *dev*.

```
{
  "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. For the `PassRole` resource, you need to provide the AWS account ID or alias, and the service role name. For more information about the IAM ARN format, see IAM ARNs in the *IAM User Guide*.

For more information specifying the null ("false") condition operator, see Condition operator to check existence of condition keys in the *IAM User Guide*.

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Sid": "PolicyCreateClusterTagNullPassroleXYZ",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:RunJobFlow"
            ],
            "Resource": [
```

```
                "*"
            ],
            "Condition": {
                "Null": {
                    "aws:RequestTag/department": "false"
                }
            }
        },
        {
            "Sid": "PolicyPassroleXYZ",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],

            "Resource": [
                "arn:aws:iam::AccountId:role/Role-Name-With-Path"
            ]
        }
    ]
}
```

Example identity-based policy statements for EMR Notebooks

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 EMR Notebooks that a user creates based on tagging

The following example policy statement, 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"
                    ]
                }
            }
        }
    ]
}
```

Denying the ModifyInstanceGroup action

The ModifyInstanceGroups action in Amazon EMR does not require that you provide a cluster ID with the action. Instead, you can specify only an instance group ID. For this reason, an apparently simple deny policy for this action based on cluster ID or a cluster tag may not have the intended effect. Consider the following example policy.

```
{
 "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups"
            ],
            "Effect": "Deny",
            "Resource": "arn:aws:elasticmapreduce:us-east-1:123456789012:cluster/
j-12345ABCDEFG67"
        }
    ]
```

```
}
```

If a user with this policy attached performs a `ModifyInstanceGroup` action and specifies only the instance group ID, the policy does not apply. Because the action is allowed on all other resources, the action is successful.

A solution to this issue is to attach a policy statement to the identity that uses a NotResource element to deny any `ModifyInstanceGroup` action issued without a cluster ID. The following example policy adds such a deny statement so that any `ModifyInstanceGroups` request fails unless a cluster ID is specified. Because an identity must specify a cluster ID with the action, deny statements based on cluster ID are therefore effective.

```
{
 "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups"
            ],
            "Effect": "Deny",
            "Resource": "arn:aws:elasticmapreduce:us-east-1:123456789012:cluster/
j-12345ABCDEFG67"
        },
        {
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups"
            ],
            "Effect": "Deny",
            "NotResource": "arn:*:elasticmapreduce:*:*:cluster/*"
        }
    ]

}
```

A similar issue exists when you want to deny the `ModifyInstanceGroups` action based on the value associated with a cluster tag. The solution is similar. In addition to a deny statement that specifies the tag value, you can add a policy statement that denies the `ModifyInstanceGroup` action if the tag that you specify is not present, regardless of value.

The following example demonstrates a policy that, when attached to an identity, denies the identity the `ModifyInstanceGroups` action any cluster with the tag `department` set to `dev`. This statement is only effective because of the deny statement that uses the `StringNotLike` condition to deny the action unless the `department` tag is present.

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

```
        {
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/department": "dev"
                }
            },
            "Effect": "Deny",
            "Resource": "*"
        },
        {
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups"
            ],
            "Condition": {
                "StringNotLike": {
                    "aws:ResourceTag/department": "?*"
                }
            },
            "Effect": "Deny",
            "Resource": "*"
        }
    ],
}
```

Authenticate to Amazon EMR cluster nodes

SSH clients can use an Amazon EC2 key pair to authenticate to cluster instances. 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. 448).

Topics

- Use an Amazon EC2 key pair for SSH credentials (p. 447)
- Use Kerberos authentication (p. 448)

Use an Amazon EC2 key pair for SSH credentials

Amazon EMR cluster nodes run on Amazon EC2 instances. You can connect to cluster nodes in the same way that you can connect to Amazon EC2 instances. You can use Amazon EC2 to create a key pair, or you can import a key pair. When you create a cluster, you can specify the Amazon EC2 key pair that will be used for SSH connections to all cluster instances. You can also create a cluster without a key pair. This is usually done with transient clusters that start, run steps, and then terminate automatically.

The SSH client that you use to connect to the cluster needs to use the private key file associated with this key pair. This is a .pem file for SSH clients using Linux, Unix and macOS. 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. 593).

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*. In addition, an EMR cluster can use an external KDC to authenticate principals.

A common scenario for establishing a cross-realm trust or using an external KDC is to authenticate users from an Active Directory domain. This allows users to access an EMR cluster with their domain user account when they use SSH to connect to a cluster or work with big data applications.

When you use Kerberos authentication, Amazon EMR configures Kerberos for the applications, components, and subsystems that it installs on the cluster so that they are authenticated with each other.

> **Important**
> Amazon EMR does not support AWS Directory Service for Microsoft Active Directory in a cross-realm trust or as an external KDC.

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.

Topics

- Supported applications (p. 448)
- Kerberos architecture options (p. 449)
- Configuring Kerberos on Amazon EMR (p. 457)
- Using SSH to connect to Kerberized clusters (p. 465)
- Tutorial: Configure a cluster-dedicated KDC (p. 466)
- Tutorial: Configure a cross-realm trust with an Active Directory domain (p. 468)

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.

The Livy user interface is the only web user interface hosted on the cluster that is Kerberized.

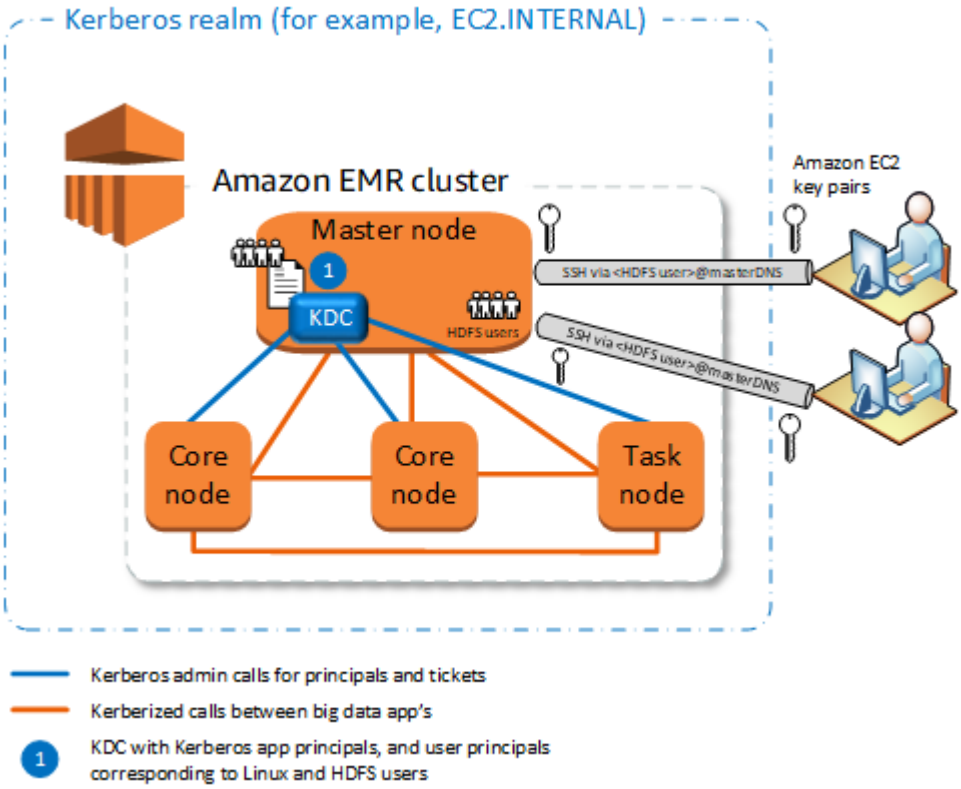
- **Hadoop MapReduce**
- **Hbase**
- **HCatalog**
- **HDFS**
- **Hive**
  - Do not enable Hive with LDAP authentication. This may cause issues communicating with Kerberized YARN.
- **Hue**
  - Hue user authentication isn't set automatically and can be configured using the configuration API.
  - Hue server is Kerberized. The Hue front-end (UI) is not configured for authentication. LDAP authentication can be configured for the Hue UI.
- **Livy**
  - Livy impersonation with Kerberized clusters is supported in Amazon EMR release version 5.22.0 and later.
- **Oozie**
- **Phoenix**
- **Spark**
- **Tez**
- **YARN**
- **Zeppelin**
  - Zeppelin is only configured to use Kerberos with the Spark interpreter. It is not configured for other interpreters.
  - User impersonation is not supported for Kerberized Zeppelin interpreters other than Spark.
- **Zookeeper**
  - Zookeeper client is not supported.

Kerberos architecture options

When you use Kerberos with Amazon EMR, you can choose from the architectures listed in this section. Regardless of the architecture that you choose, you configure Kerberos using the same steps. You create a security configuration, you specify the security configuration and compatible cluster-specific Kerberos options when you create the cluster, and you create HDFS directories for Linux users on the cluster that match user principals in the KDC. For an explanation of configuration options and example configurations for each architecture, see Configuring Kerberos on Amazon EMR (p. 457).

Cluster-dedicated KDC (KDC on master node)

This configuration is available with Amazon EMR release version 5.10.0 and later.



Advantages

- Amazon EMR has full ownership of the KDC.
- The KDC on the EMR cluster is independent from centralized KDC implementations such as Microsoft Active Directory or AWS Managed Microsoft AD.
- Performance impact is minimal because the KDC manages authentication only for local nodes within the cluster.
- Optionally, other Kerberized clusters can reference the KDC as an external KDC. For more information, see External KDC—master node on a different cluster (p. 454).

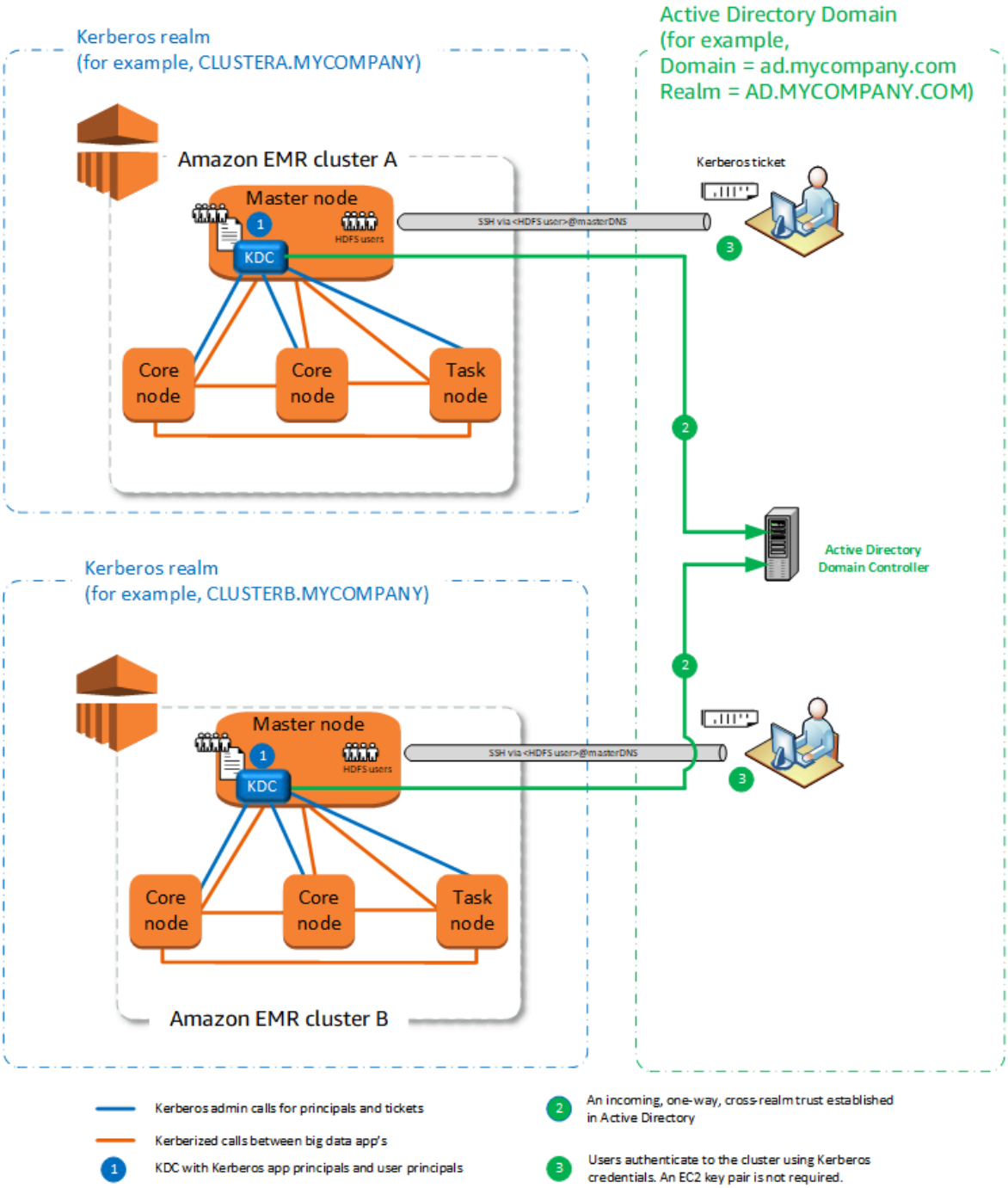
Considerations and limitations

- Kerberized clusters can not authenticate to one another, so applications can not interoperate. If cluster applications need to interoperate, you must establish a cross-realm trust between clusters, or set up one cluster as the external KDC for other clusters. If a cross-realm trust is established, the KDCs must have different Kerberos realms.
- You must create Linux users on the EC2 instance of the master node that correspond to KDC user principals, along with the HDFS directories for each user.
- User principals must use an EC2 private key file and `kinit` credentials to connect to the cluster using SSH.

Cross-realm trust

In this configuration, principals (usually users) from a different Kerberos realm authenticate to application components on a Kerberized EMR cluster, which has its own KDC. The KDC on the master node 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 in each KDC. Cross-realm trusts are most common with Active Directory implementations, as shown in the following diagram. Cross-realm trusts with an external MIT KDC or a KDC on another Amazon EMR cluster are also supported.



Advantages

- The EMR cluster on which the KDC is installed maintains full ownership of the KDC.
- With Active Directory, Amazon EMR automatically creates Linux users that correspond to user principals from the KDC. You still must create HDFS directories for each user. In addition, user

principals in the Active Directory domain can access Kerberized clusters using `kinit` credentials, without the EC2 private key file. This eliminates the need to share the private key file among cluster users.

- Because each cluster KDC manages authentication for the nodes in the cluster, the effects of network latency and processing overhead for a large number of nodes across clusters is minimized.

Considerations and limitations

- If you are establishing a trust with an Active Directory realm, you must provide an Active Directory user name and password with permissions to join principals to the domain when you create the cluster.
- Cross-realm trusts cannot be established between Kerberos realms with the same name.
- Cross-realm trusts must be established explicitly. For example, if Cluster A and Cluster B both establish a cross-realm trust with a KDC, they do not inherently trust one another and their applications cannot authenticate to one another to interoperate.
- KDCs must be maintained independently and coordinated so that credentials of user principals match precisely.

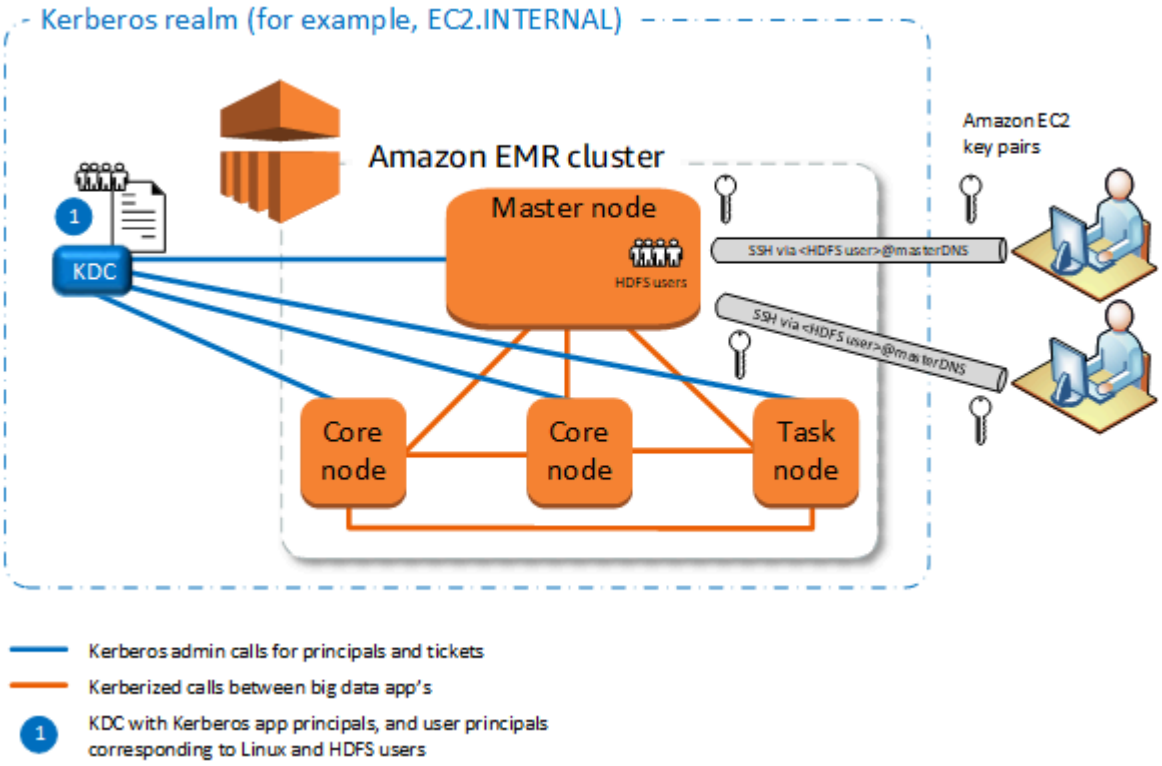
External KDC

Configurations with an External KDC are supported with Amazon EMR 5.20.0 and later.

- External KDC—MIT KDC (p. 452)
- External KDC—master node on a different cluster (p. 454)
- External KDC—cluster KDC on a different cluster with Active Directory cross-realm trust (p. 455)

External KDC—MIT KDC

This configuration allows one or more EMR clusters to use principals defined and maintained in an MIT KDC server.



Advantages

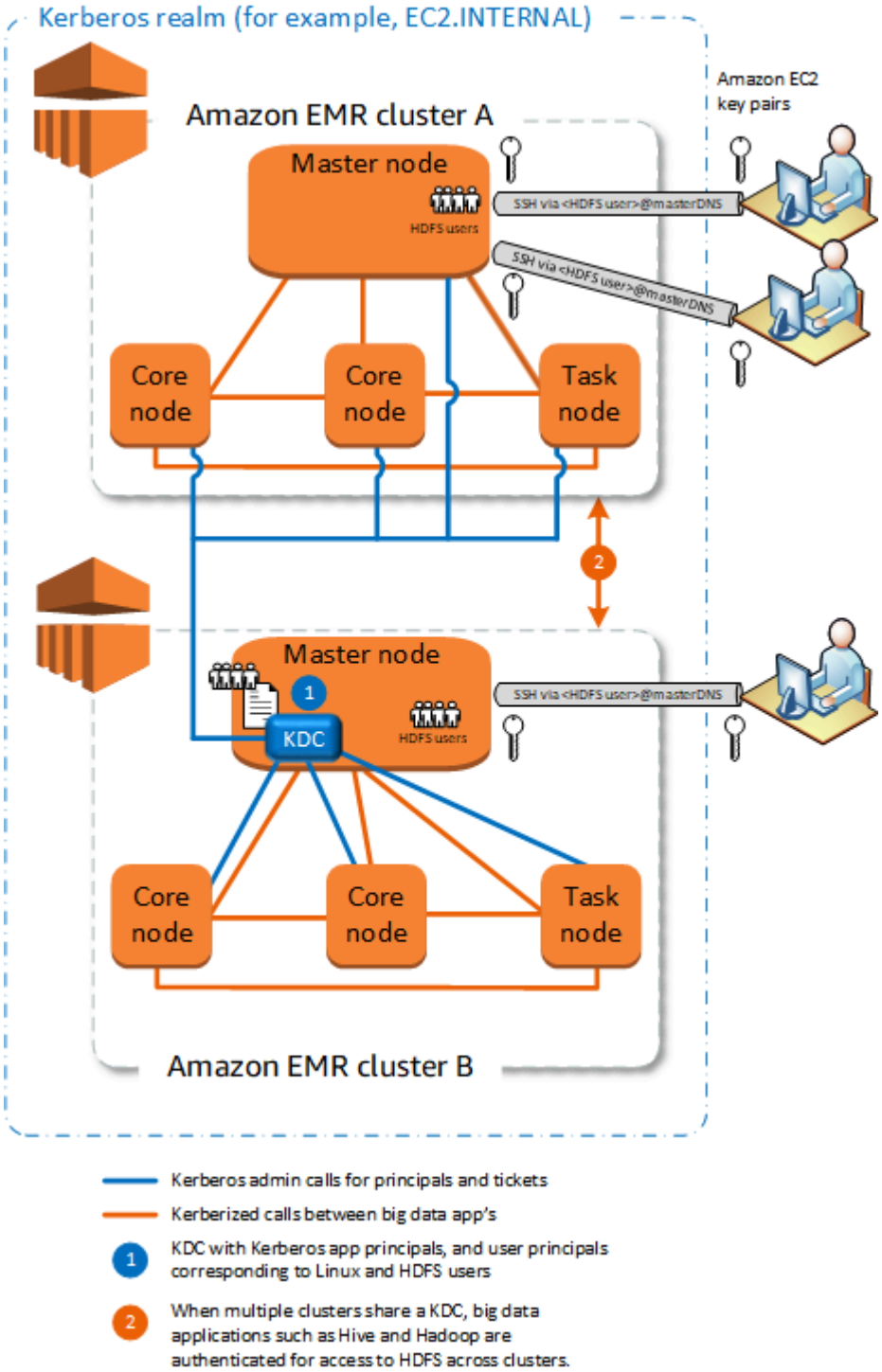
- Managing principals is consolidated in a single KDC.
- Multiple clusters can use the same KDC in the same Kerberos realm. This allows cluster applications to interoperate and simplifies the authentication of communication between clusters as compared to a cross-realm trust.
- The master node on a Kerberized cluster does not have the performance burden associated with maintaining the KDC.

Considerations and limitations

- You must create Linux users on the EC2 instance of each Kerberized cluster's master node that correspond to KDC user principals, along with the HDFS directories for each user.
- User principals must use an EC2 private key file and `kinit` credentials to connect to Kerberized clusters using SSH.
- Each node in Kerberized EMR clusters must have a network route to the KDC.
- Each node in Kerberized clusters places an authentication burden on the external KDC, so the configuration of the KDC affects cluster performance. When you configure the hardware of the KDC server, consider the maximum number of Amazon EMR nodes to be supported simultaneously.
- Cluster performance is dependent on the network latency between nodes in Kerberized clusters and the KDC.
- Troubleshooting can be more difficult because of interdependencies.

External KDC—master node on a different cluster

This configuration is nearly identical to the external MIT KDC implementation above, except that the KDC is on the master node of an EMR cluster. For more information, see Cluster-dedicated KDC (KDC on master node) (p. 449) and Tutorial: Configure a cross-realm trust with an Active Directory domain (p. 468).



Advantages

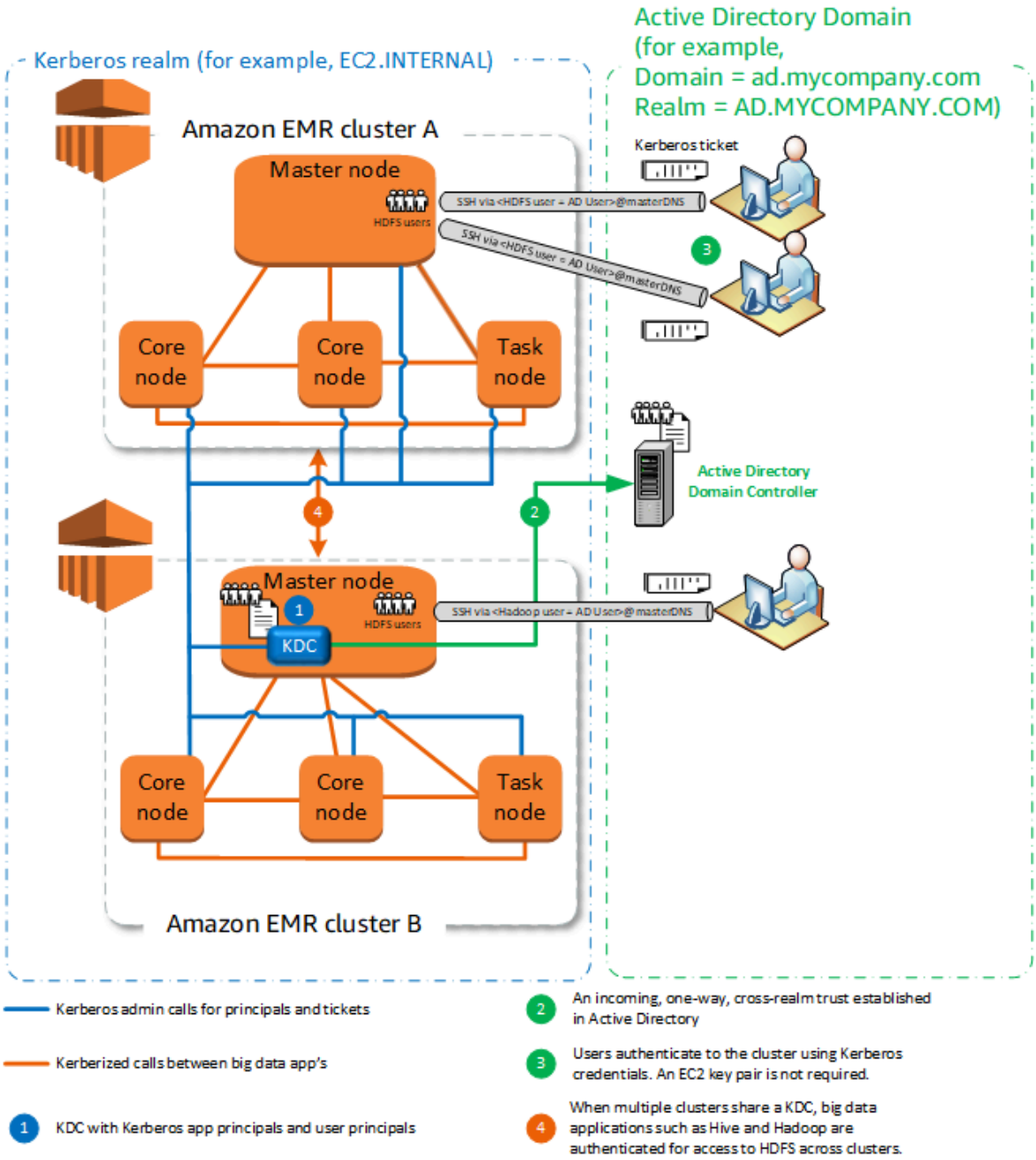
- Managing principals is consolidated in a single KDC.
- Multiple clusters can use the same KDC in the same Kerberos realm. This allows cluster applications an Kerberized clusters to interoperate. It also simplifies the authentication of communication between clusters as compared to a cross-realm trust.

Considerations and limitations

- You must create Linux users on the EC2 instance of each Kerberized cluster's master node that correspond to KDC user principals, along with the HDFS directories for each user.
- User principals must use an EC2 private key file and `kinit` credentials to connect to Kerberized clusters using SSH.
- Each node in each EMR cluster must have a network route to the KDC.
- Each Amazon EMR node in Kerberized clusters places an authentication burden on the external KDC, so the configuration of the KDC affects cluster performance. When you configure the hardware of the KDC server, consider the maximum number of Amazon EMR nodes to be supported simultaneously.
- Cluster performance is dependent on the network latency between nodes in the clusters and the KDC.
- Troubleshooting can be more difficult because of interdependencies.

External KDC—cluster KDC on a different cluster with Active Directory cross-realm trust

In this configuration, you first create a cluster with a cluster-dedicated KDC that has a one-way cross-realm trust with Active Directory. For a detailed tutorial, see Tutorial: Configure a cross-realm trust with an Active Directory domain (p. 468). You then launch additional clusters, referencing the cluster KDC that has the trust as an external KDC. For an example, see External cluster KDC with Active Directory cross-realm trust (p. 462). This allows each Amazon EMR cluster that uses the external KDC to authenticate principals defined and maintained in a Microsoft Active Directory domain.



Advantages

- Managing principals is consolidated in the Active Directory domain.
- Amazon EMR joins the Active Directory realm, which eliminates the need to create Linux users that correspond Active Directory users. You still must create HDFS directories for each user.
- Multiple clusters can use the same KDC in the same Kerberos realm, which is different from the Active Directory realm. This allows cluster applications to interoperate.
- User principals in the Active Directory domain can access Kerberized clusters using `kinit` credentials, without the EC2 private key file. This eliminates the need to share the private key file among cluster users.

- Only one Amazon EMR master node has the burden of maintaining the KDC, and only that cluster must be created with Active Directory credentials for the cross-realm trust between the KDC and Active Directory.

Considerations and limitations

- Each node in each EMR cluster must have a network route to the KDC and the Active Directory domain controller.
- Each Amazon EMR node places an authentication burden on the external KDC, so the configuration of the KDC affects cluster performance. When you configure the hardware of the KDC server, consider the maximum number of Amazon EMR nodes to be supported simultaneously.
- Cluster performance is dependent on the network latency between nodes in the clusters and the KDC server.
- Troubleshooting can be more difficult because of interdependencies.

Configuring Kerberos on Amazon EMR

This section provides configuration details and examples for setting up Kerberos with common architectures. Regardless of the architecture you choose, the configuration basics are the same and done in three steps. If you use an external KDC or set up a cross-realm trust, you must ensure that every node in a cluster has a network route to the external KDC, including the configuration of applicable security groups to allow inbound and outbound Kerberos traffic.

Step 1: Create a security configuration with Kerberos properties

The security configuration specifies details about the Kerberos KDC, and allows the Kerberos configuration to be re-used each time you create a cluster. You can create a security configuration 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 about creating security configurations and specifying a security configuration when you create a cluster, see Use security configurations to set up cluster security (p. 355). For information about Kerberos properties in a security configuration, see Kerberos settings for security configurations (p. 458).

Step 2: Create a cluster and specify cluster-specific Kerberos attributes

When you create a cluster, you specify a Kerberos security configuration along with cluster-specific Kerberos options. When you use the Amazon EMR console, only the Kerberos options compatible with the specified security configuration are available. When you use the AWS CLI or Amazon EMR API, ensure that you specify Kerberos options compatible with the specified security configuration. For example, if you specify a principal password for a cross-realm trust when you create a cluster using the CLI, and the specified security configuration is not configured with cross-realm trust parameters, an error occurs. For more information, see Kerberos settings for clusters (p. 460).

Step 3: Configure the cluster master node

Depending on the requirements of your architecture and implementation, additional set up on the cluster may be required. You can do this after you create it or using steps or bootstrap actions during the creation process.

For each Kerberos-authenticated user that connects to the cluster using SSH, you must ensure that Linux user accounts are created that correspond to the Kerberos user. If user principals are provided by an Active Directory domain controller, either as the external KDC or through a cross-realm trust, Amazon EMR creates Linux user accounts automatically. If Active Directory is not used, you must create principals for each user that correspond to their Linux user. For more information, see Configuring a cluster for Kerberos-authenticated HDFS users and SSH connections (p. 463).

Each user also must also have an HDFS user directory that they own, which you must create. In addition, SSH must be configured with GSSAPI enabled to allow connections from Kerberos-authenticated users. GSSAPI must be enabled on the master node, and the client SSH application must be configured to use GSSAPI. For more information, see Configuring a cluster for Kerberos-authenticated HDFS users and SSH connections (p. 463).

Security configuration and cluster settings for Kerberos on Amazon EMR

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.

This topic provides an overview of the configuration parameters available for Kerberos when you create a security configuration and a cluster. In addition, CLI examples for creating compatible security configurations and clusters are provided for common architectures.

Kerberos settings for security configurations

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. 355).

Use the following references to understand the available security configuration settings for the Kerberos architecture that you choose. Amazon EMR console settings are shown. For corresponding CLI options, see Specifying Kerberos settings using the AWS CLI (p. 366) or Configuration examples (p. 461).

Parameter		Description
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.
Provider	**Cluster-dedicated KDC**	Specifies that Amazon EMR creates a KDC on the master node of any cluster that uses this security configuration. You specify the realm name and KDC admin password when you create the cluster. You can reference this KDC from other clusters, if required. Create those clusters using a different security configuration, specify an external KDC, and use the realm name and KDC admin password that you specify for the cluster-dedicated KDC.
	External KDC	Available only with Amazon EMR 5.20.0 and later. Specifies that clusters using this security configuration authenticate Kerberos principals using a KDC server outside the cluster. A KDC is not created on the cluster. When you create the cluster, you specify the realm name and KDC admin password for the external KDC.
Ticket Lifetime		Optional. Specifies the period for which a Kerberos ticket issued by the KDC is valid on clusters that use this security configuration. 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 `kinit` from the master node command line to renew after a ticket expires.

Parameter		Description
Cross-realm trust		Specifies a cross-realm trust between a cluster-dedicated KDC on clusters that use this security configuration and a KDC in a different Kerberos realm. Principals (typically users) from another realm are authenticated to clusters that use this configuration. Additional configuration in the other Kerberos realm is required. For more information, see Tutorial: Configure a cross-realm trust with an Active Directory domain (p. 468).
Cross-realm trust properties	**Realm**	Specifies the Kerberos realm name of the other realm in the trust relationship. By convention, Kerberos realm names are the same as the domain name 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) or IP address 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) or IP address 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 use 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`).
External KDC		Specifies that clusters external KDC is used by the cluster.
External KDC properties	**Admin server**	Specifies the fully qualified domain name (FQDN) or IP address of the external admin server. 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`).

Parameter			Description
	KDC server		Specifies the fully qualified domain name (FQDN) of the external KDC server. The KDC server and admin server typically run on the same machine with the same FQDN, but use 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`).
	Active Directory Integration		Specifies that Kerberos principal authentication is integrated with a Microsoft Active Directory domain.
	Active Directory integration properties	**Active Directory realm**	Specifies the Kerberos realm name of the Active Directory domain. By convention, Kerberos realm names are typically the same as the domain name but in all capital letters.
		Active Directory domain	Specifies the Active Directory domain name.
		Active Directory server	Specifies the fully qualified domain name (FQDN) of the Microsoft Active Directory domain controller.

Kerberos settings for clusters

You can specify Kerberos settings when you create a cluster using the Amazon EMR console, the AWS CLI, or the EMR API.

Use the following references to understand the available cluster configuration settings for the Kerberos architecture that you choose. Amazon EMR console settings are shown. For corresponding CLI options, see Configuration examples (p. 461).

Parameter	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 `ec2.internal`, using `EC2.INTERNAL` as the realm name.
KDC admin password	The password used within the cluster for `kadmin` or `kadmin.local`. 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.
Active Directory domain join user (optional)	Required when using Active Directory in a cross-realm trust. This is the user logon name of an Active Directory account with permission to join computers to the domain. Amazon EMR uses this

Parameter	Description
	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. 470).
Active Directory domain join password (optional)	The password for the Active Directory domain join user. For more information, see the section called "Step 3: Add user accounts to the domain for the EMR Cluster" (p. 470).

Configuration examples

The following examples demonstrate security configurations and cluster configurations for common scenarios. AWS CLI commands are shown for brevity.

Local KDC

The following commands create a cluster with a cluster-dedicated KDC running on the master node. Additional configuration on the cluster is required. For more information, see Configuring a cluster for Kerberos-authenticated HDFS users and SSH connections (p. 463).

Create Security Configuration

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

Create Cluster

```
aws emr create-cluster --release-label emr-5.36.0 \
--instance-count 3 --instance-type m5.xlarge \
--applications Name=Hadoop Name=Hive --ec2-attributes
 InstanceProfile=EMR_EC2_DefaultRole,KeyName=MyEC2Key \
--service-role EMR_DefaultRole \
--security-configuration LocalKDCSecurityConfig \
--kerberos-attributes Realm=EC2.INTERNAL,KdcAdminPassword=MyPassword
```

Cluster-dedicated KDC with Active Directory cross-realm trust

The following commands create a cluster with a cluster-dedicated KDC running on the master node with a cross-realm trust to an Active Directory domain. Additional configuration on the cluster and in Active Directory is required. For more information, see Tutorial: Configure a cross-realm trust with an Active Directory domain (p. 468).

Create Security Configuration

```
aws emr create-security-configuration --name LocalKDCWithADTrustSecurityConfig \
--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"}}}}}'
```

Create Cluster

```
aws emr create-cluster --release-label emr-5.36.0 \
--instance-count 3 --instance-type m5.xlarge --applications Name=Hadoop Name=Hive \
--ec2-attributes InstanceProfile=EMR_EC2_DefaultRole,KeyName=MyEC2Key \
--service-role EMR_DefaultRole --security-configuration KDCWithADTrustSecurityConfig \
--kerberos-attributes Realm=EC2.INTERNAL,KdcAdminPassword=MyClusterKDCAdminPassword,\
ADDomainJoinUser=ADUserLogonName,ADDomainJoinPassword=ADUserPassword,\
CrossRealmTrustPrincipalPassword=MatchADTrustPassword
```

External KDC on a different cluster

The following commands create a cluster that references a cluster-dedicated KDC on the master node of a different cluster to authenticate principals. Additional configuration on the cluster is required. For more information, see Configuring a cluster for Kerberos-authenticated HDFS users and SSH connections (p. 463).

Create Security Configuration

```
aws emr create-security-configuration --name ExtKDCOnDifferentCluster \
--security-configuration '{"AuthenticationConfiguration": \
{"KerberosConfiguration": {"Provider": "ExternalKdc", \
"ExternalKdcConfiguration": {"KdcServerType": "Single", \
"AdminServer": "MasterDNSOfKDCMaster:749", \
"KdcServer": "MasterDNSOfKDCMaster:88"}}}}'
```

Create Cluster

```
aws emr create-cluster --release-label emr-5.36.0 \
--instance-count 3 --instance-type m5.xlarge \
--applications Name=Hadoop Name=Hive \
--ec2-attributes InstanceProfile=EMR_EC2_DefaultRole,KeyName=MyEC2Key \
--service-role EMR_DefaultRole --security-configuration ExtKDCOnDifferentCluster \
--kerberos-attributes Realm=EC2.INTERNAL,KdcAdminPassword=KDCOnMasterPassword
```

External cluster KDC with Active Directory cross-realm trust

The following commands create a cluster with no KDC. The cluster references a cluster-dedicated KDC running on the master node of another cluster to authenticate principals. That KDC has a cross-realm trust with an Active Directory domain controller. Additional configuration on the master node with the KDC is required. For more information, see Tutorial: Configure a cross-realm trust with an Active Directory domain (p. 468).

Create Security Configuration

```
aws emr create-security-configuration --name ExtKDCWithADIntegration \
--security-configuration '{"AuthenticationConfiguration": \
{"KerberosConfiguration": {"Provider": "ExternalKdc", \
"ExternalKdcConfiguration": {"KdcServerType": "Single", \
"AdminServer": "MasterDNSofClusterKDC:749", \
"KdcServer": "MasterDNSofClusterKDC.com:88", \
"AdIntegrationConfiguration": {"AdRealm":"AD.DOMAIN.COM", \
"AdDomain":"ad.domain.com", \
"AdServer":"ad.domain.com"}}}}}'
```

Create Cluster

```
aws emr create-cluster --release-label emr-5.36.0 \
--instance-count 3 --instance-type m5.xlarge --applications Name=Hadoop Name=Hive \
--ec2-attributes InstanceProfile=EMR_EC2_DefaultRole,KeyName=MyEC2Key \
--service-role EMR_DefaultRole --security-configuration ExtKDCWithADIntegration \
```

```
--kerberos-attributes Realm=EC2.INTERNAL,KdcAdminPassword=KDCOnMasterPassword,\
ADDomainJoinUser=MyPrivilegedADUserName,ADDomainJoinPassword=PasswordForADDomainJoinUser
```

Configuring a cluster for Kerberos-authenticated HDFS users and SSH connections

Amazon EMR creates Kerberos-authenticated user clients for the 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 then connect to the cluster with their Kerberos credentials and work with applications. For a user to authenticate to the cluster, the following configurations are required:

- A Linux user account matching the Kerberos principal in the KDC must exist on the cluster. Amazon EMR does this automatically in architectures that integrate with Active Directory.
- You must create an HDFS user directory on the master node for each user, and give the user permissions to the directory.
- You must configure the SSH service so that GSSAPI is enabled on the master node. In addition, users must have an SSH client with GSSAPI enabled.

Adding Linux users and Kerberos principals to the master node

If you do not use Active Directory, you must create Linux accounts on the cluster master node and add principals for these Linux users to the KDC. This includes a principal in the KDC for the master node. In addition to the user principals, the KDC running on the master node needs a principal for the local host.

When your architecture includes Active Directory integration, Linux users and principals on the local KDC, if applicable, are created automatically. You can skip this step. For more information, see Cross-realm trust (p. 450) and External KDC—cluster KDC on a different cluster with Active Directory cross-realm trust (p. 455).

> **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.

Submitting a step to the cluster after you create it or when you create the cluster is the easiest way to add users and KDC principals. Alternatively, you can connect to the master node using an EC2 key pair as the default `hadoop` user to run the commands. For more information, see Connect to the master node using SSH (p. 593).

The following example submits a bash script `configureCluster.sh` to a cluster that already exists, referencing its cluster ID. The script is saved to Amazon S3.

```
aws emr add-steps --cluster-id <j-2AL4XXXXXX5T9> \
--steps Type=CUSTOM_JAR,Name=CustomJAR,ActionOnFailure=CONTINUE,\
Jar=s3://region.elasticmapreduce/libs/script-runner/script-runner.jar,\
Args=["s3://DOC-EXAMPLE-BUCKET/configureCluster.sh"]
```

The following example demonstrates the contents of the `configureCluster.sh` script. The script also handles creating HDFS user directories and enabling GSSAPI for SSH, which are covered in the following sections.

```
#!/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=([lijuan]=pwd1 [marymajor]=pwd2 [richardroe]=pwd3)
for i in ${!arr[@]}; do
    #Assign plain language variables for clarity
     name=${i}
     password=${arr[${i}]}

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

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

     #Change owner of each user's hdfs directory to that 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 systemctl restart sshd
```

Adding user HDFS directories

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.

Submitting a step to the cluster after you create it or when you create the cluster is the easiest way to create HDFS directories. Alternatively, you could connect to the master node using an EC2 key pair as the default hadoop user to run the commands. For more information, see Connect to the master node using SSH (p. 593).

The following example submits a bash script `AddHDFSUsers.sh` to a cluster that already exists, referencing its cluster ID. The script is saved to Amazon S3.

```
aws emr add-steps --cluster-id <j-2AL4XXXXXX5T9> \
--steps Type=CUSTOM_JAR,Name=CustomJAR,ActionOnFailure=CONTINUE,\
Jar=s3://region.elasticmapreduce/libs/script-runner/script-runner.jar,Args=["s3://DOC-
EXAMPLE-BUCKET/AddHDFSUsers.sh"]
```

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

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

# Initialize an array of user names from AD, or Linux users created manually on the cluster
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
```

Enabling GSSAPI for SSH

For Kerberos-authenticated users to connect to the master node using SSH, the SSH service must have GSSAPI authentication enabled. To enable GSSAPI, run the following commands from the master node command line or use a step to run it as a script. After reconfiguring SSH, you must restart the service.

```
sudo sed -i 's/^.*GSSAPIAuthentication.*$/GSSAPIAuthentication yes/' /etc/ssh/sshd_config
sudo sed -i 's/^.*GSSAPICleanupCredentials.*$/GSSAPICleanupCredentials yes/' /etc/ssh/
sshd_config
sudo systemctl restart sshd
```

Using SSH to connect to Kerberized clusters

This section demonstrates the steps for a Kerberos-authenticated user to connect to the master node of an EMR cluster.

Each computer that is used for an SSH connection must have SSH client and Kerberos client applications installed. Linux computers most likely include these 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. Windows users can use applications such as PuTTY as an SSH client.

For more information about SSH connections, see Connect to the cluster (p. 591).

SSH uses GSSAPI for authenticating Kerberos clients, and you must enable GSSAPI authentication for the SSH service on the cluster master node. For more information, see Enabling GSSAPI for SSH (p. 465). SSH clients must also use GSSAPI.

In the following examples, 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`*.

Prerequisite for krb5.conf (non-Active Directory)

When using a configuration without Active Directory integration, in addition to the SSH client and Kerberos client applications, each client computer must have a copy of the `/etc/krb5.conf` file that matches the `/etc/krb5.conf` file on the cluster master node.

To copy the krb5.conf file

1. Use SSH to connect to the master node using an EC2 key pair and the default `hadoop` user —for example, `hadoop@`*`MasterPublicDNS`*. For detailed instructions, see Connect to the cluster (p. 591).
2. From the master node, copy the contents of the `/etc/krb5.conf` file . For more information, see Connect to the cluster (p. 591).
3. On each client computer that will connect to the cluster, create an identical `/etc/krb5.conf` file based on the copy that you made in the previous step.

Using kinit and SSH

Each time a user connects from a client computer using Kerberos credentials, the user must first renew Kerberos tickets for their user on the client computer. In addition, the SSH client must be configured to use GSSAPI authentication.

To use SSH to connect to a Kerberized EMR cluster

1. Use `kinit` to renew your Kerberos tickets as shown in the following example

```
kinit user1
```

2. Use an `ssh` client along with the principal that you created in the cluster-dedicated KDC or Active Directory user name. Make sure that GSSAPI authentication is enabled as shown in the following examples.

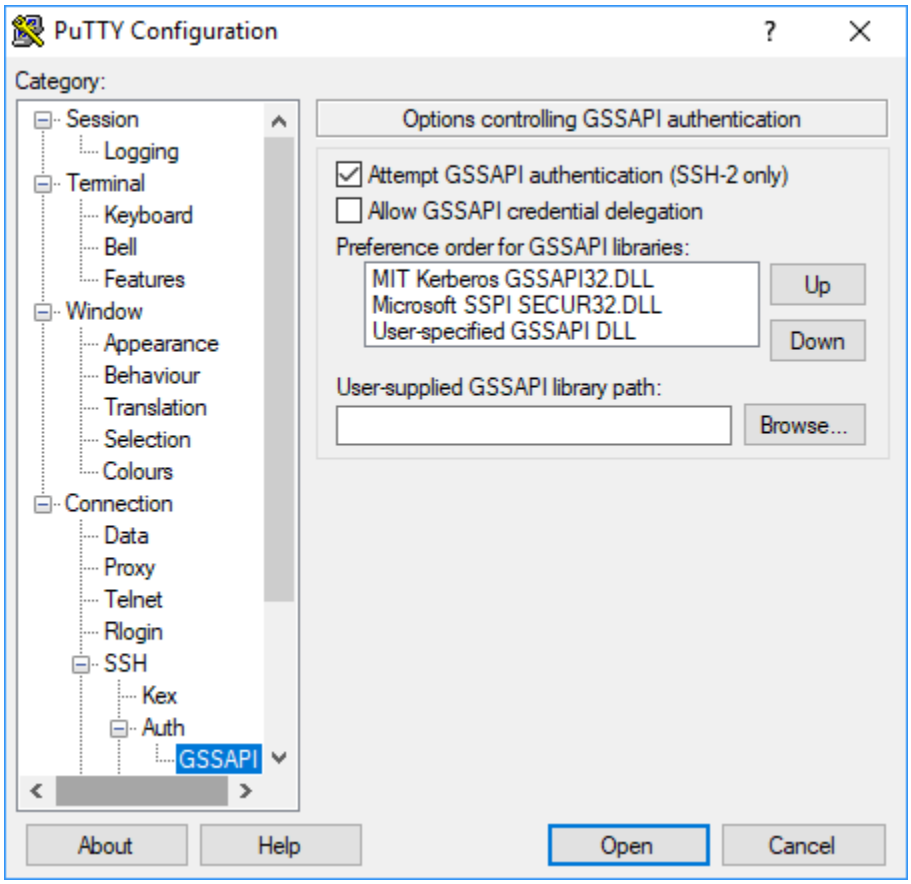
   Example: Linux users

   The `-K` option specifies GSSAPI authentication.

```
ssh -K user1@MasterPublicDNS
```

   Example: Windows users (PuTTY)

   Make sure that the GSSAPI authentication option for the session is enabled as shown:



Tutorial: Configure a cluster-dedicated KDC

This topic guides you through creating a cluster with a cluster-dedicated *key distribution center (KDC)*, 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.

For more information on Amazon EMR support for Kerberos and KDC, as well as links to MIT Kerberos Documentation, see Use Kerberos authentication (p. 448).

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.

```
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-5.36.0 \
--instance-type m5.xlarge \
--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://DOC-EXAMPLE-BUCKET/createlinuxusers.sh
```

The following code 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 can 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-2AL4XXXXXX5T9> --steps
 Type=CUSTOM_JAR,Name=CustomJAR,ActionOnFailure=CONTINUE,Jar=s3://
```

```
myregion.elasticmapreduce/libs/script-runner/script-runner.jar,Args=["s3://DOC-EXAMPLE-
BUCKET/configurekdc.sh"]
```

The following code 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 systemctl restart sshd
```

The users that you added should now be able to connect to the cluster using SSH. For more information, see Using SSH to connect to Kerberized clusters (p. 465).

Tutorial: Configure a cross-realm trust with an Active Directory domain

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 *key distribution center (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 one another 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. Any network setup that allows the required network traffic between KDCs is acceptable.

Optionally, after you establish a cross-realm trust with Active Directory using a KDC on one cluster, you can create another cluster using a different security configuration to reference the KDC on the first cluster as an external KDC. For an example security configuration and cluster set up, see External cluster KDC with Active Directory cross-realm trust (p. 462).

For more information on Amazon EMR support for Kerberos and KDC, as well as links to MIT Kerberos Documentation, see Use Kerberos authentication (p. 448).

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

Step 1: Set up the VPC and subnet (p. 469)

Step 2: Launch and install the Active Directory domain controller (p. 470)

Step 3: Add user accounts to the domain for the EMR Cluster (p. 470)

Step 4: Configure an incoming trust on the Active Directory domain controller (p. 470)

Step 5: Use a DHCP option set to specify the Active Directory domain controller as a VPC DNS server (p. 471)

Step 6: Launch a Kerberized EMR Cluster (p. 471)

Step 7: Create HDFS users and set permissions on the cluster for Active Directory user accounts (p. 472)

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. 471).

The KDC and the Active Directory domain controller must be able to resolve one other's domain names. 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. Make a note of the Group ID of the security group associated with the EC2 instance. You need it for Step 6: Launch a Kerberized EMR Cluster (p. 471). We use *`sg-012xrlmdomain345`*. Alternatively, you can specify different security groups for the EMR cluster and this instance that allows traffic between them. For more information, see Amazon EC2 security groups for Linux instances in the *Amazon EC2 User Guide for Linux Instances*.
3. 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*.
4. 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. 471). 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`*. Replace the *`KDC-FQDN`* with the **Public DNS** name listed for the Amazon EMR master node hosting the KDC. 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 KDC-FQDN
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.internal` 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. 470).

```
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 with the following attributes:
   - Use the `--security-configuration` option to specify the security configuration that you created. We use *`MyKerberosConfig`* in the example.
   - Use the `SubnetId` property of the `--ec2-attributes` option to specify the subnet that you created in Step 1: Set up the VPC and subnet (p. 469). We use *`step1-subnet`* in the example.
   - Use the `AdditionalMasterSecurityGroups` and `AdditionalSlaveSecurityGroups` of the `--ec2-attributes` option to specify that the security group associated with the AD domain controller from Step 2: Launch and install the Active Directory domain controller (p. 470) is associated with the cluster master node as well as core and task nodes. We use *`sg-012xrlmdomain345`* in the example.

   Use `--kerberos-attributes` to 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. 470).
   - 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. 470).

The following example launches a Kerberized cluster.

```
aws emr create-cluster --name "MyKerberosCluster" \
--release-label emr-5.10.0 \
--instance-type m5.xlarge \
--instance-count 3 \
--ec2-attributes InstanceProfile=EMR_EC2_DefaultRole,KeyName=MyEC2KeyPair,\
SubnetId=step1-subnet, AdditionalMasterSecurityGroups=sg-012xrlmdomain345,
AdditionalSlaveSecurityGroups=sg-012xrlmdomain345\
--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,\
ADDomainJoinUser=ADUserLogonName,ADDomainJoinPassword=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. 447).

Run the following command to add a step to the cluster that runs a script, *`AddHDFSUsers.sh`*.

```
aws emr add-steps --cluster-id <j-2AL4XXXXXX5T9> \
--steps Type=CUSTOM_JAR,Name=CustomJAR,ActionOnFailure=CONTINUE,\
Jar=s3://region.elasticmapreduce/libs/script-runner/script-runner.jar,Args=["s3://DOC-
EXAMPLE-BUCKET/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 or Linux users and KDC principals created
 manually on the cluster
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 Using SSH to connect to Kerberized clusters (p. 465), 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.

Integrate Amazon EMR with AWS Lake Formation

Beginning with Amazon EMR 5.31.0, you can launch a cluster that integrates with AWS Lake Formation. AWS Lake Formation is a managed service that helps you discover, catalog, cleanse, and secure data in an Amazon Simple Storage Service (Amazon S3) data lake. For more information, see AWS Lake Formation.

This section provides a conceptual overview of Amazon EMR integration with Lake Formation. It also lists the prerequisites and steps required to launch an Amazon EMR cluster integrated with Lake Formation.

Integrating Amazon EMR with AWS Lake Formation provides the following key benefits:

- Fine-grained, column-level access to databases and tables in the AWS Glue Data Catalog.
- Federated single sign-on to EMR Notebooks or Apache Zeppelin from enterprise identity systems compatible with Security Assertion Markup Language (SAML) 2.0.

> **Important**
> If you currently use EMR clusters with Lake Formation in beta mode, you should upgrade your clusters to EMR version 5.31.0 or above to continue using this feature. Clusters with an EMR version below 5.31.0 will stop working with Lake Formation.
>
> **Note**
> EMR integration with Lake Formation is not yet available for the EMR 6.x series and does not currently support using AWS IAM Identity Center (successor to AWS Single Sign-On) for federated single sign-on.

Topics

- Overview of Amazon EMR integration with Lake Formation (p. 473)
- Applications, features, and limitations (p. 479)
- Before you begin (p. 480)
- Launch an Amazon EMR cluster with Lake Formation (p. 489)

Overview of Amazon EMR integration with Lake Formation

When you integrate Amazon EMR with AWS Lake Formation, you enable SAML-based authentication using corporate credentials for EMR Notebooks and Apache Zeppelin, and can enforce fine-grained, column-level access control to data lakes based on policies you define in AWS Lake Formation.

Requirements

Your organization must meet the following requirements before you integrate Amazon EMR and Lake Formation:

- Manage your corporate identities with an existing, SAML-based identity provider (IdP) such as Active Directory Federation Services (AD FS). For more information, see Configure third-party providers for SAML (p. 484).
- Use the AWS Glue Data Catalog as a metadata store.
- Define and manage permissions in Lake Formation to access databases, tables, and columns in AWS Glue Data Catalog. For more information, see AWS Lake Formation.
- Use EMR Notebooks or Apache Zeppelin to access data managed by AWS Glue and Lake Formation.

Integration steps

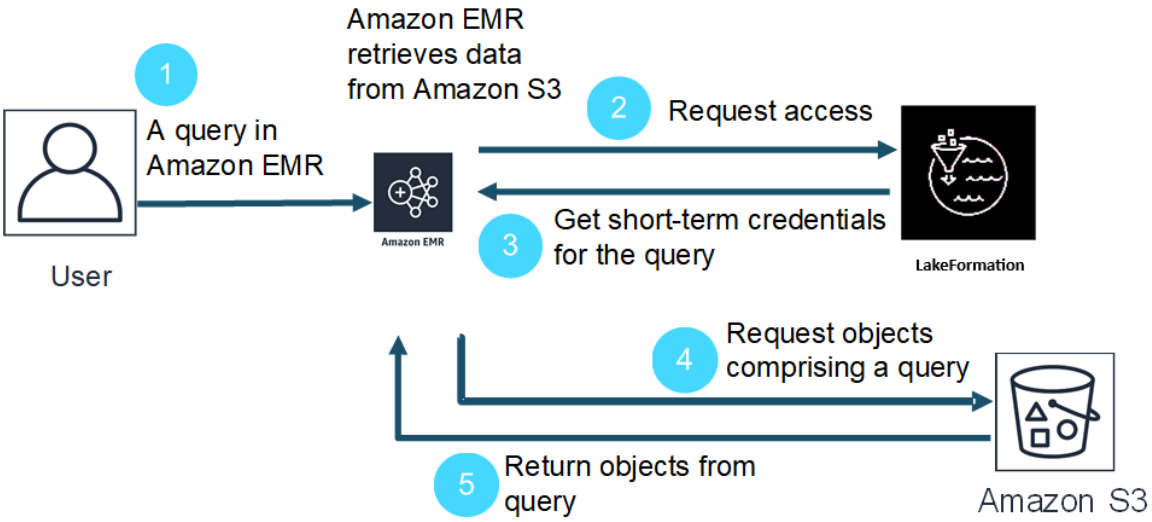
To integrate Amazon EMR with Lake Formation, you will need to take the following steps:

1. Complete prerequisite tasks including configuring your identity provider, creating IAM roles for Lake Formation, setting up a security configuration, and preparing Lake Formation resources. For more information about prerequisites for Amazon EMR and Lake Formation integration, see Before you begin (p. 480).
2. Launch a cluster with the new roles and security configuration you created for Lake Formation. For more information, see Launch an Amazon EMR cluster with Lake Formation (p. 489).
3. Update the callback or Single Sign-on URL with your IdP to redirect users after successful SAML authentication to the master node of your cluster. For more information, see Update the callback or single sign-on URL with your Identity Provider (p. 490).

How data access works in Lake Formation

After you integrate Amazon EMR with Lake Formation, users authenticate through your organization's Identity Provider (IdP) sign-on page to access EMR Notebooks or Zeppelin. Then, Lake Formation provides access to data through temporary credentials to EMR. This process is known as credential vending. For more information, see AWS Lake Formation.

Following is a high-level overview of how EMR gets access to data protected by Lake Formation security policies:



1. A user submits a query in Amazon EMR on data from Lake Formation.
2. Amazon EMR requests temporary credentials from AWS Lake Formation for access for that user.

3. Lake Formation returns temporary credentials, allowing data access.
4. Amazon EMR sends the query request to retrieve data from Amazon S3.
5. Amazon EMR receives the data from Amazon S3, then filters and returns the results based on the user permissions that you defined in Lake Formation.

For detailed information about how user authentication and access to data works, see Amazon EMR components (p. 477) and Architecture of SAML-enabled single sign-on and fine-grained access control (p. 478).

For more information about adding users and groups to Lake Formation policies, see Granting Data Catalog permissions.

Topics

- IAM roles for Lake Formation (p. 475)
- Terms and concepts (p. 477)
- Amazon EMR components (p. 477)
- Architecture of SAML-enabled single sign-on and fine-grained access control (p. 478)

IAM roles for Lake Formation

The integration between Amazon EMR and AWS Lake Formation relies on three key roles you should create before launching your cluster:

1. A custom Amazon EC2 instance profile for Amazon EMR.
2. An IAM role for Lake Formation use for identity federation.
3. An IAM role for non-Lake Formation AWS services.

This section gives an overview of these roles and the policies that you need to include for each role. For information about creating these roles, see Before you begin (p. 480).

EC2 instance profile

Amazon EMR uses IAM service roles to perform actions on your behalf to provision and manage clusters. The service role for cluster EC2 instances, also called the EC2 instance profile for Amazon EMR, is a special type of service role assigned to every EC2 instance in a cluster during launch.

To define permissions for EMR clusters to interact with Lake Formation and other AWS services, you should define a custom EC2 instance profile to use instead of the `EMR_EC2_DefaultRole` when you launch your cluster.

For more information, see Service role for cluster EC2 instances (EC2 instance profile) and Customize IAM roles.

IAM role for Lake Formation

The IAM Role for Lake Formation defines what privileges a user logging in through your IdP will have and which identity provider (IdP) can assume this role. The role's `Maximum CLI/API session duration` defines the session timeout for access to EMR Notebooks and Apache Zeppelin.

- This role must be created with the following permissions policy.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "lakeformation:GetDataAccess",
      "glue:GetUnfiltered*",
      "glue:GetTable",
      "glue:GetTables",
      "glue:GetDatabase",
      "glue:GetDatabases",
      "glue:CreateDatabase",
      "glue:GetUserDefinedFunction",
      "glue:GetUserDefinedFunctions",
      "glue:GetPartition",
      "glue:GetPartitions"
    ],
    "Resource": "*"
  }
}
```

Note

Do not grant this role permissions to access any Amazon S3 buckets managed by AWS Glue. The federated user should access data through Lake Formation using Spark SQL and should not access data directly through Amazon S3.

- The role must also include the following trust policy, which allows your IAM identity provider to assume the role. Replace *account-id* with your AWS account ID. Replace the *IAM_identity_provider_name* with your IAM identity provider's name.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::account-id:saml-
provider/IAM_identity_provider_name"
            },
            "Action": "sts:AssumeRoleWithSAML"
        }
    ]
}
```

IAM role for non-Lake Formation AWS services

The IAM Role for AWS Services defines the permissions that the Amazon EMR cluster has when accessing non-AWS Lake Formation services. For example, if the jobs running on your cluster require access to Amazon DynamoDB, or any other AWS services, the IAM role for AWS Services should have policies required to access those services.

When you configure policies for this role, make sure the role does not have access to the following API operations:

- Any AWS Glue API operations.
- Any AWS Lake Formation API operations.
- Any AWS Security Token Service (STS) `AssumeRole` operations.
- Any Amazon S3 access to buckets managed by AWS Glue. The cluster should access data through Lake Formation using Spark SQL and should not access data directly through Amazon S3.

Terms and concepts

This section provides definitions for concepts and terms that are used in the context of integrating Amazon EMR with AWS Lake Formation.

Authentication

The process of establishing the identity of a user. By integrating Amazon EMR and Lake Formation, your users can use their corporate credentials to log into EMR Notebooks and Apache Zeppelin.

Authorization

The process of permitting which actions a given user can take on a resource. When integrating an Amazon EMR cluster with Lake Formation, you define policies that authorize user access to databases, tables, and columns. This process ensures users may only query and analyze the tables or columns for which they're authorized.

Federation

The creation of a trust relationship between an external identity provider (IdP) and AWS Identity and Access Management(IAM). Users can sign in through a compatible, Security Assertion Markup Language (SAML) 2.0 identity provider (IdP), such as Microsoft Active Directory Federation Services. For more information, see Configure third-party providers for SAML (p. 484). When you use SAML 2.0 to configure a trust relationship between an IdP and IAM, AWS assigns the user an IAM role. The user also receives temporary credentials that allow the user to access your AWS Lake Formation resources.

Trust policy

A document in JSON format that defines who is allowed to assume an IAM role. The document is written according to the rules of the IAM policy language.

Permissions policy

A permissions document in JSON format that defines the actions and resources to which an IAM role has access. The document is written according to the rules of the IAM policy language.

Principal

An entity that can access resources protected by Lake Formation policies and run queries in Amazon EMR. Principals are federated users.

Amazon EMR components

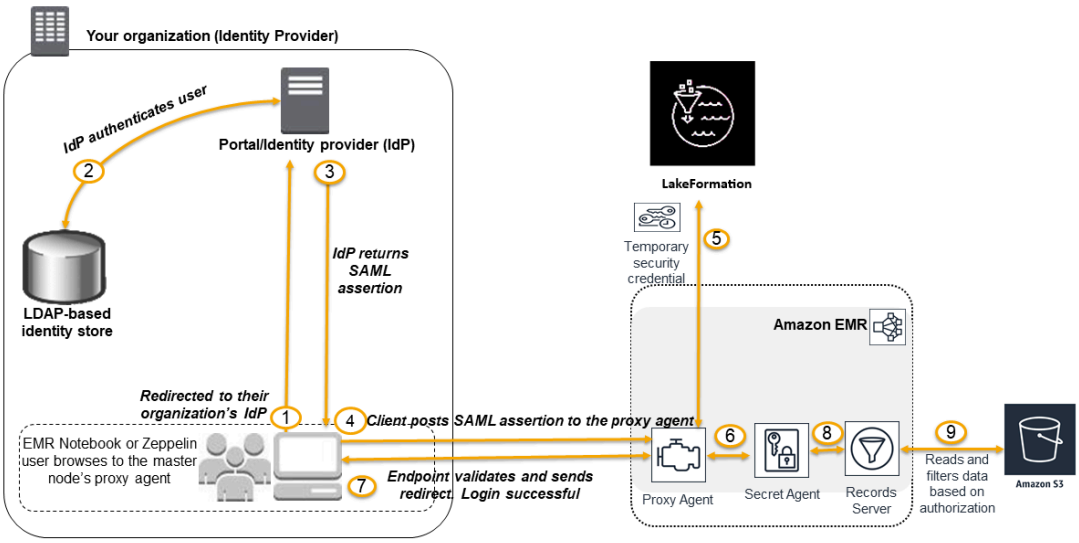
Amazon EMR enables fine-grained access control with Lake Formation through the following components:

- **Proxy agent** - The proxy agent is based on Apache Knox. It receives SAML-authenticated requests from users and translates SAML claims to temporary credentials. It also stores the temporary credentials in the secret agent. The proxy agent runs on the master node as the `knox` system user and writes logs to the `/var/log/knox` directory.
- **Secret agent** - The secret agent securely stores secrets and distributes secrets to other EMR components or applications. The secrets can include temporary user credentials, encryption keys, or Kerberos tickets. The secret agent runs on every node in the cluster and uses Lake Formation and AWS Glue APIs to retrieve temporary credentials and AWS Glue Data Catalog metadata. The secret agent runs as the `emrsecretagent` user, and writes logs to the `/emr/secretagent/log` directory. The process relies on a specific set of `iptables` rules to function. It is important to ensure `iptables` is not disabled, and, if you customize `iptables` configuration, the `nat` table rules must be preserved and left unaltered.
- **Record server** - The record server receives requests to access data. It then authorizes requests based on temporary credentials and table access control policies distributed by the secret agent. The record

server reads data from Amazon S3 and returns column-level data that the user is authorized to access. The record server runs on every node in the cluster as the `emr_record_server` user and writes logs to the `/var/log/emr-record-server` directory.

Architecture of SAML-enabled single sign-on and fine-grained access control

The following diagram illustrates the architecture of SAML-enabled single sign-on and fine-grained access control with Lake Formation and Amazon EMR.



1. An unauthenticated user uses the proxy agent to access EMR notebook or Zeppelin. The user is redirected to your organization's Identity Provider (IdP) sign-on page.
2. The IdP verifies the user's identity in your organization.
3. The IdP generates a SAML authentication response that includes assertions that identify the user and include attributes about the user.
4. The client browser posts the SAML assertion to the proxy agent.
5. The proxy agent requests user-specific temporary security credentials from AWS Lake Formation on behalf of the user. The temporary security credentials are sent back to the proxy agent.
6. The proxy agent stores the user-specific temporary security credentials in the secret agent. The secret agent sends the temporary user credentials to the secret agents in core and task nodes.
7. The proxy agent enables successful user login.
8. When the user runs a Spark job by using the EMR Notebooks or Zeppelin, the record server calls the secret agent to obtain temporary user credentials.
9. The record server reads and filters data from Amazon S3 based on the policies defined in Lake Formation.

From the user's perspective, this process happens transparently. The user starts at your organization's authentication page and ends up at the EMR Notebooks or Zeppelin interface through the browser without ever having to supply any AWS credentials.

Applications, features, and limitations

Supported applications

The integration between Amazon EMR and AWS Lake Formation supports the following applications:

- EMR Notebooks
- Apache Zeppelin
- Apache Spark through EMR Notebooks

> **Important**
> Other applications are currently not supported. To ensure the security of your cluster, do not install applications that do not appear in this list.

Supported features

The following Amazon EMR features can be used with EMR and Lake Formation:

- Encryption at rest and in transit
- Kerberos authentication using a cluster-dedicated key distribution center (KDC) without cross-realm trust
- Instance groups, instance fleets, and Spot Instances
- Reconfiguring applications on a running cluster
- Monitoring data access using AWS CloudTrail

  > **Note**
  > AWS CloudTrail entries are created when notebook users access data protected by Lake Formation with Spark SQL. Each CloudTrail entry displays the Active Directory name of the user in the `lakeFormationPrincipal` property.
- EMRFS server-side encryption (SSE)

  > **Note**
  > Amazon EMR encryption settings govern SSE. For more information, see Encryption options (p. 374).

The following EMR features currently do not work with Lake Formation integration:

- Steps
- Multiple master nodes
- EMRFS consistent view
- EMRFS client-side encryption (CSE)

Limitations

Consider the following limitations when using Amazon EMR with AWS Lake Formation:

- Amazon EMR with Lake Formation is currently available in 16 AWS Regions: US East (Ohio and N. Virginia), US West (N. California and Oregon), Asia Pacific (Mumbai, Seoul, Singapore, Sydney, and Tokyo), Canada (Central), Europe (Frankfurt, Ireland, London, Paris, and Stockholm), South America (São Paulo).
- Amazon EMR with Lake Formation does not currently work with AWS IAM Identity Center (successor to AWS Single Sign-On) (SSO).

- You must use a user-defined IAM role instead of the Lake Formation service-linked role to register data locations used by Amazon EMR clusters with Lake Formation. Lake Formation does not support using its service-linked role when you integrate with EMR. For information about creating a user-defined role to register data locations with Lake Formation, see Requirements for roles used to register locations.
- It is important to understand that the Lake Formation column-level authorization prevents users from accessing data in columns that the user does not have access to. However, in certain situations, users are able to access metadata describing all columns in the table, including the columns the user does not have access to. This column metadata is stored in the table's table properties for tables that use the Avro storage format or that use custom Serializer/Deserializers (SerDe) in which the table schema is defined in table properties along with the SerDe definition. When you use Amazon EMR and Lake Formation, we recommend that you review the contents of the table properties for the tables you're protecting and, where possible, limit the information stored in table properties to prevent any sensitive metadata from being visible to users.
- In Lake Formation enabled clusters, users can only access data managed by AWS Glue Data Catalog. Users cannot access data managed outside of AWS Glue or Lake Formation. Data from other sources can be accessed using non-Spark SQL operations if the IAM role for other AWS Services chosen during cluster deployment has policies in place allowing the cluster to access those data sources.

  For example, you might have two Amazon S3 buckets and an Amazon DynamoDB table that you want your Spark job to access in addition to a set of Lake Formation tables. In this case, you could create a role that can access the two Amazon S3 buckets, and the Amazon DynamoDB table and use it for the `IAM role for other AWS services` when launching your cluster.
- Spark job submission must be done through EMR Notebooks, Zeppelin, or Livy. Spark jobs submitted through `spark-submit` will not work with Lake Formation at this time.
- Using Spark SQL to access Lake Formation tables that use the Hive Map data type is currently not supported.
- There is currently no central logout available for EMR Notebooks and Zeppelin.
- Spark's fallback to HDFS for statistics collection capability is not supported in this release with Lake Formation. The property `spark.sql.statistics.fallBackToHdfs` for this feature is disabled by default. This feature does not work when this property is manually enabled.
- Querying tables that contain partitions under different table paths in Amazon S3 is currently not supported.
- Cross-realm trust for Kerberos authentication is currently not supported.

Before you begin

Before you launch an Amazon EMR cluster with AWS Lake Formation, complete the following tasks:

1. Set up a trust relationship between your identity provider and AWS to enable SAML 2.0-based federation, and create IAM roles for Lake Formation. For instructions, see Configure a trust relationship between your IdP and Lake Formation (p. 481).
2. Create a new Amazon EC2 instance profile. For instructions, see Create a customized Amazon EC2 instance profile (p. 486).
3. Configure Amazon EMR security features. For instructions, see Configure EMR security (p. 487).

You should also complete the following AWS Lake Formation tasks covered in the *AWS Lake Formation Developer Guide*:

1. Allow data filtering for data lakes on Amazon EMR by opting in. You may opt in before or after launching an Amazon EMR cluster with Lake Formation, but you must explicitly allow data filtering before Amazon EMR can access data in Amazon S3 locations registered with Lake Formation. For more information and instructions, see Allow data filtering on Amazon EMR in the *Lake Formation Developer Guide*.

2. Create a user-defined service role for Lake Formation to register data locations that will be accessed by Amazon EMR. For instructions, see Requirements for roles used to register locations.

   Warning
   You must use a user-defined role and not the Lake Formation service-linked role when you register data locations. Lake Formation does not support using its service-linked role when you integrate with EMR.
3. Set up and control user access to resources through Lake Formation policies in the AWS Lake Formation console. For more information, see Lake Formation permissions.

Topics

- Configure a trust relationship between your IdP and Lake Formation (p. 481)
- Configure third-party providers for SAML (p. 484)
- Create a customized Amazon EC2 instance profile (p. 486)
- Configure EMR security (p. 487)

Configure a trust relationship between your IdP and Lake Formation

To establish a trust relationship between your organization's Identity Provider (IdP) and AWS, you must do the following:

- Tell your IdP about AWS as a service provider.
- Tell AWS about your external IdP by creating an IAM identity provider and role for SAML access in AWS Identity and Access Management.

This process is called adding a *relying party trust*. For more information about relying party trusts, see Configuring your SAML 2.0 IdP with relying party trust and adding claims in the AWS Identity and Access Management User Guide.

To add a relying party trust between Lake Formation and your IdP

1. Register AWS with your IdP. The process of registering AWS with your IdP depends on which IdP you use. For more information on how to do this for Auth0, Microsoft Active Directory Federation Services (AD FS), and Okta, see Configure third-party providers for SAML (p. 484).
2. Using your IdP, generate a metadata XML file that can describe your IdP as an IAM identity provider in AWS. Your IdP metadata XML file must include the following items:

   - Issuer name
   - Creation date
   - Expiration date
   - Keys that AWS will use to validate authentication responses (assertions) from your organization.

   Each IdP has a specific way of simply exporting this metadata. For more information, refer to the documentation for your IdP.

   Upload the metadata XML file to an Amazon S3 bucket. When you launch a cluster that integrates with Lake Formation, you will specify the path to the S3 bucket.
3. Create a SAML Identity Provider.

   a. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

b. In the navigation pane, choose **Identity Providers**, **Create Provider**.

c. For **Provider Type**, choose **Choose a provider type**, **SAML**.

d. Enter a name for the identity provider.

e. For **Metadata Document**, click **Choose File**, specify the SAML metadata document that you downloaded from your IdP in the previous step, and choose **Open**.

f. Verify the information that you have provided, and click **Create**.

4. Create an IAM role for Lake Formation.

a. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

b. In the navigation pane, choose **Roles**, **Create role**.

c. Choose the **SAML 2.0 federation** role type.

d. For **SAML Provider**, choose the provider for your role.

e. Choose **Allow programmatic and AWS Management Console access** to create a role that can be assumed programmatically and from the console.

f. Review your SAML 2.0 trust information, then choose **Next: Permissions**.

g. Create the permissions policy for the role based on the example below. For more information about this permissions policy, see IAM roles for Lake Formation (p. 475).

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "lakeformation:GetDataAccess",
      "glue:GetUnfiltered*",
      "glue:GetTable",
      "glue:GetTables",
      "glue:GetDatabase",
      "glue:GetDatabases",
      "glue:CreateDatabase",
      "glue:GetUserDefinedFunction",
      "glue:GetUserDefinedFunctions"
    ],
    "Resource": "*"
  }
}
```

Note

Do not grant this role permissions to access any Amazon S3 buckets managed by AWS Glue. The federated user should access data through Lake Formation using Spark SQL and should not access data directly through Amazon S3.

h. Choose **Next: Tags**.

i. Choose **Next: Review**.

j. For **Role name**, type a role name. Role names must be unique within your AWS account.

k. Review the role and then choose **Create role**.

l. Click Roles tab, search for the role name created from the last step.

m. Choose **Trust relationships**, and then select **Edit trust relationship**.

n. Override the existing policy document with the IAM Role for Lake Formation trust policy specified below. For more information about this trust policy, see IAM roles for Lake Formation (p. 475) section.

Replace *account-id* with your AWS account ID. Replace the *IAM_identity_provider_name* with your IAM identity provider's name.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::account-id:saml-
provider/IAM_identity_provider_name"
            },
            "Action": "sts:AssumeRoleWithSAML"
        }
    ]
}
```

o. Choose **Update Trust Relationship**.

5. Create an IAM role for other AWS services. For more information, see IAM role for non-Lake Formation AWS services (p. 476).
6. In your organization's IdP, you must configure SAML assertions that map the users in your organization to the IdP and the IAM role for Lake Formation that you just created. Do this by configuring the three attribute elements shown in the following table.

   - Replace *account-id* with your AWS account ID.
   - Replace *IAM_Role_For_Lake_Formation* with the name of the IAM role for Lake Formation that you created.
   - Replace *IAM_identity_provider_name* with the name of the IAM identity provider that you created in the previous steps.
   - Replace *user_alias* with the name of the attribute used to hold the user name defined in your organization.

Attribute elements	Value
`https://aws.amazon.com/SAML/Attributes/Role`	arn:aws:iam::*account-id*:role/*IAM_Role_For_Lake_Formation*,arn:aws:iam::*account-id*:saml-provider/*IAM_identity_provider_name*
`https://aws.amazon.com/SAML/Attributes/RoleSessionName`	*user_alias*
`https://lakeformation.amazon.com/SAML/Attributes/Username`	*user_alias*

The exact steps to perform this mapping depend on which IdP you use. For instructions on how to map your IdP, see Configure third-party providers for SAML (p. 484).

For more information, see Configuring SAML assertions for the authentication response.

Note

A user can get Lake Formation privileges for all the groups they belong to. The list of groups a user belongs to can be sent in a custom SAML attribute that is specific to Lake Formation, as the following example demonstrates.

```
<AttributeStatement>
<Attribute Name="https://lakeformation.amazon.com/SAML/Attributes/Groups">
<AttributeValue>group1</AttributeValue>
<AttributeValue>group2</AttributeValue>
</Attribute>
</AttributeStatement>
```

Configure third-party providers for SAML

SAML 2.0-based federation for Amazon EMR with AWS Lake Formation has been tested with Auth0, Microsoft Active Directory Federation Services (AD FS), and Okta.

This topic provides information to help you configure the tested identity providers to work with AWS Lake Formation federation. If you decide to use another identity provider (IdP) that has not been tested for AWS Lake Formation, refer to the documentation website for your IdP for more information.

Auth0

The AWS integration in Auth0 page on the Auth0 documentation website describes how to set up single sign-on (SSO) with the AWS Management Console. It also includes a JavaScript example.

To enable federated access to Lake Formation, modify the following steps in the Auth0 documentation:

- When providing an application callback URL, provide a temporary URL, as shown in the following example. Update *public-dns* with the actual DNS name for your master node after launching your cluster.

```
https://public-dns:8442/gateway/knoxsso/api/v1/websso?
pac4jCallback=true&client_name=SAML2Client
```

- When configuring SAML, paste the following SAML configuration code into **Settings**.

```
{
    "audience": "urn:amazon:webservices",
    "mappings": {
        "email": "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress",
        "name": "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name"
    },
    "createUpnClaim": false,
    "passthroughClaimsWithNoMapping": false,
    "mapUnknownClaimsAsIs": false,
    "mapIdentities": false,
    "nameIdentifierFormat": "urn:oasis:names:tc:SAML:2.0:nameid-format:persistent",
    "nameIdentifierProbes": [
        "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"
    ]
}
```

- When mapping the AWS role to a user, create a rule with the following code. Replace the *IAM_Role_For_Lake_Formation* with the name of the IAM role for Lake Formation that you created. Replace the *IAM_identity_provider_name* with the name of the IAM identity provider that you created for Auth0.

```
function (user, context, callback) {
  user.awsRole = 'arn:aws:iam::account-
id:role/IAM_Role_For_Lake_Formation,arn:aws:iam::account-id:saml-
provider/IAM_identity_provider_name';
  // the username must not contain "@" - as it is not a valid Linux username
  user.userName = user.name.replace(/@.*/, '');
```

```
  context.samlConfiguration.mappings = {
    'https://aws.amazon.com/SAML/Attributes/Role': 'awsRole',
    'https://aws.amazon.com/SAML/Attributes/RoleSessionName': 'userName',
    'https://lakeformation.amazon.com/SAML/Attributes/Username': 'userName'
  };

  callback(null, user, context);
}
```

Microsoft Active Directory Federation Services (AD FS)

The AWS federated authentication with Active Directory Federation Services (AD FS) AWS Security Blog shows how to configure AD FS and enable SAML federation with AWS.

To enable federated access to Lake Formation, modify the following steps in the blog post:

- To add relying party trust, manually enter data about the relying party instead of importing metadata from the existing URL. Select the **Permit all users to access this relying party** option. For the endpoint trusted URL, provide a temporary URL, as shown in the following example. Update *public-dns* with the actual DNS name for your master node after launching your cluster.

```
https://public-dns:8442/gateway/knoxsso/api/v1/websso?
pac4jCallback=true&client_name=SAML2Client
```

- In the step of **Edit Claim Issuance Policy**, customize the three rules `NameId`, `RoleSessionName`, and `Role` based on the values for the attribute elements in Configure a trust relationship between your IdP and Lake Formation (p. 481).

Okta

The Set up a SAML application in Okta page on the Okta support site includes instructions on how to configure Okta by providing metadata about the relying party.

To enable federated access to Lake Formation, modify the following steps from the Okta support site:

- When configuring SAML, for the **Single sign-on URL**, use the temporary URL, as shown in the following example. Update the *public-dns* with the actual DNS name for your master node after launching your cluster.

```
https://public-dns:8442/gateway/knoxsso/api/v1/websso?
pac4jCallback=true&client_name=SAML2Client
```

- For the **Audience URI (SP Entity ID)** box, fill in `urn:amazon:webservices`.
- In the **Attribute Statements** section, add three attribute statements as demonstrated in the following procedure. Replace the *IAM_Role_For_Lake_Formation* with the name of the IAM role for Lake Formation that you created. Replace the *IAM_identity_provider_name* with the name of the IAM identity provider that you created in previous steps. Replace *user_alias* with the name of the attribute used to hold the user name defined in your organization.
  1. Name: `https://aws.amazon.com/SAML/Attributes/Role`

     Value: `arn:aws:iam::`*account-id*`:role/`*IAM_Role_For_Lake_Formation*`,arn:aws:iam::`*account-id*`:saml-provider/`*IAM_identity_provider_name*
  2. Name: `https://aws.amazon.com/SAML/Attributes/RoleSessionName`

     Value: *user_alias*

3. Name: `https://lakeformation.amazon.com/SAML/Attributes/Username`

   Value: *`user_alias`*

Create a customized Amazon EC2 instance profile

To orchestrate EMR cluster access to Lake Formation, you should define a customized EC2 instance profile (service role) that you will specify when you create your cluster. For more information about creating a customized role, see Creating a service role for cluster EC2 instances with least-privilege permissions (p. 405).

The `lakeformation:GetTemporaryUserCredentialsWithSAML` action lets your instance profile obtain the necessary credentials to run queries on Amazon EMR clusters that are integrated with AWS Lake Formation.

Public subnet

If your cluster will be in a public subnet, use the following example policy for your customized EC2 instance profile. Replace *`My-S3-Bucket-for-IdP-Metadata`* with the S3 bucket containing your identify provider (IdP) metadata. Replace the *`MyEMRClusterLogs`* with an S3 bucket for storing EMR logs.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::My-S3-Bucket-for-IdP-Metadata/*"
            ],
            "Action": "s3:GetObject"
        },
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::My-S3-Bucket-for-IdP-Metadata"
            ],
            "Action": "s3:ListBucket"
        },
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::MyEMRClusterLogs/*"
            ],
            "Action": "s3:Put*"
        },
        {
            "Effect": "Allow",
            "Resource": "*",
            "Action": "lakeformation:GetTemporaryUserCredentialsWithSAML"
        }
  {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:PassRole"
            ],
            "Resource": "*"
        }
    ]
}
```

Private subnet

If your cluster will be in a private subnet, use the following example policy for your customized EC2 instance profile. Replace *My-S3-Bucket-for-IdP-Metadata* with the S3 bucket containing your identify provider (IdP) metadata. Replace the *MyEMRClusterLogs* with the S3 bucket for storing EMR logs.

When using Amazon EMR clusters in a private subnet, make sure your EMR cluster can access the AWS Lake Formation service to fetch required metadata and query policies and the AWS IAM service to extract IAM role session duration value. To access AWS Lake Formation service, you can either set up a NAT Gateway or go through VPC Endpoint for AWS Lake Formation. To access AWS IAM service, you need to set up a NAT Gateway, as currently AWS IAM does not support VPC Endpoint. Note that you will incur cost when using either NAT Gateway or VPC Endpoint service.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::My-S3-Bucket-for-IdP-Metadata/*",
                "arn:aws:s3:::packages.*.amazonaws.com/*",
                "arn:aws:s3:::repo.*.amazonaws.com/*",
            ],
            "Action": "s3:GetObject"
        },
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::My-S3-Bucket-for-IdP-Metadata"
            ],
            "Action": "s3:ListBucket"
        },
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::MyEMRClusterLogs/*"
            ],
            "Action": "s3:Put*"
        },
        {
            "Effect": "Allow",
            "Resource": "*",
            "Action": "lakeformation:GetTemporaryUserCredentialsWithSAML"
        }
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:PassRole"
            ],
            "Resource": "*"
        }
    ]
}
```

Configure EMR security

Create an Amazon EMR security configuration for Lake Formation

Before you launch an Amazon EMR cluster integrated with AWS Lake Formation, you'll need to create a security configuration with the IAM roles and identity provider (IdP) metadata XML file you created

in Configure a trust relationship between your IdP and Lake Formation (p. 481). You will specify this security configuration when you launch the cluster.

Console

To create a security configuration that specifies the AWS Lake Formation integration option:

1. In the Amazon EMR console, select **Security configurations**, then **Create**.
2. Type a **Name** for the security configuration. You will use this name to specify the security configuration when you create a cluster.
3. Under **AWS Lake Formation integration**, select **Enable fine-grained access control managed by AWS Lake Formation**.
4. Select your **IAM role for AWS Lake Formation** to apply.

   Note
   For more information, see IAM roles for Lake Formation (p. 475).
5. Select your **IAM role for other AWS services** to apply.
6. Upload your identify provider (IdP) metadata by specifying the S3 path where the metadata is located.

   Note
   For more information, see Configure a trust relationship between your IdP and Lake Formation (p. 481).
7. Set up other security configuration options as appropriate and choose **Create**. You must enable Kerberos authentication using the cluster-dedicated KDC. For more information, see Configure EMR security (p. 487).

CLI

To create a security configuration for AWS Lake Formation integration:

1. Specify the whole path to your IdP metadata file uploaded in S3.
2. Replace *account-id* with your AWS account ID.
3. Specify a value for `TicketLifetimeInHours` to determine the period for which a Kerberos ticket issued by the KDC is valid.

```
{
    "LakeFormationConfiguration": {
        "IdpMetadataS3Path": "s3://mybucket/myfolder/idpmetadata.xml",
        "EmrRoleForUsersARN": "arn:aws:iam::account-
id:role/IAM_Role_For_AWS_Services",
        "LakeFormationRoleForSAMLPrincipalARN": "arn:aws:iam::account-
id:role/IAM_Role_For_Lake_Formation"
    },
    "AuthenticationConfiguration": {
        "KerberosConfiguration": {
            "Provider": "ClusterDedicatedKdc",
            "ClusterDedicatedKdcConfiguration": {
                "TicketLifetimeInHours": 24
            }
        }
    }
}
```

4. Create an Amazon EMR security configuration with the following command. Replace *security-configuration* with a name of your choice. You will select this configuration by name when you create your cluster.

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

Configure additional security features

To securely integrate Amazon EMR with AWS Lake Formation, you should also configure the following EMR security features:

- Enable Kerberos authentication using the cluster-dedicated KDC. For instructions, see Use Kerberos authentication.
- Configure your Amazon EC2 security group or Amazon VPC network access control list (ACL) to allow access to the proxy agent (port 8442) from your user's desktops. For more information, see Control network traffic with security groups.
- (Optional) Enable encryption in transit or at rest. For more information, see Encryption options in the *Amazon EMR Management Guide*.
- (Optional) Create a custom Transport Layer Security (TLS) key pair for the proxy agent. For more information, see Customize your proxy agent certificate (p. 492).

For more information, see Security in Amazon EMR.

Launch an Amazon EMR cluster with Lake Formation

This section provides information about how to launch an Amazon EMR cluster integrated with AWS Lake Formation. It also shows you how to update the single sign-on URL in your identity provider (IdP), how to use EMR Notebooks with Lake Formation, how to customize your proxy agent certificate, and how to set up cross-account Lake Formation access for your cluster.

Topics

- Launch an Amazon EMR cluster with Lake Formation (p. 489)
- Set up EMR Notebooks to access Lake Formation data (p. 491)
- Set up cross-account access (p. 493)

Launch an Amazon EMR cluster with Lake Formation

Before you launch an Amazon EMR cluster with AWS Lake Formation, you must complete the prerequisites in Before you begin (p. 480).

When you launch an Amazon EMR cluster with Lake Formation, you must specify the following items.

- A **supported Amazon EMR release label**. EMR with Lake Formation works with the Amazon EMR 5.x series, starting with **5.31.0**. EMR integration with Lake Formation is not yet available for the 6.x series.
- Your **security configuration** for Lake Formation. If you need to create a security configuration, see Configure EMR security (p. 487). For instructions on how to select a security configuration when you create a cluster, see Specify a security configuration for a cluster (p. 373).
- Your **customized EC2 instance profile**. To set up a customized Amazon EC2 instance profile, see Create a customized Amazon EC2 instance profile (p. 486). For information about how to select your instance profile when you launch a cluster, see Specify custom IAM roles when you create a cluster (p. 418).
- Your **Kerberos settings**. For more information, see Kerberos settings for clusters (p. 460).

After you launch your cluster, ensure you update the callback or single sign-on URL with your identity provider (p. 490) to direct users back to the master node of the cluster.

Example: Create an EMR Cluster with Lake Formation using the AWS CLI

The following AWS CLI command launches an Amazon EMR cluster in a public subnet with Zeppelin integrated with AWS Lake Formation. It includes specified values for `--kerberos-attributes`, `--security-configuration`, and `InstanceProfile` as required for EMR integration with Lake Formation.

```
aws emr create-cluster --region us-east-1 \
--release-label emr-5.31.0 \
--use-default-roles \
--instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m4.xlarge \
InstanceGroupType=CORE,InstanceCount=1,InstanceType=m4.xlarge \
--applications Name=Zeppelin Name=Livy \
--kerberos-attributes Realm=EC2.INTERNAL,KdcAdminPassword=MyClusterKDCAdminPassword \
--ec2-attributes
 KeyName=EC2_KEY_PAIR,SubnetId=subnet-00xxxxxxxxxxxx11,InstanceProfile=MyCustomEC2InstanceProfile
 \
--security-configuration security-configuration \
--name cluster-name
```

If you're launching your cluster in a private subnet without a Network Address Translation (NAT) Gateway attached, you'll need to add both a AWS Glue Data Catalog and interface VPC endpoints in order for your cluster to integrate with Lake Formation. For more information, see AWS Lake Formation and interface VPC endpoints (AWS PrivateLink).

Update the callback or single sign-on URL with your Identity Provider

After you launch an Amazon EMR cluster with Lake Formation, you must update the callback or single-sign-on URL with your identity provider in order to direct users back to the master node of your cluster.

1. Locate the public IP address of the master node and the master instance ID for your cluster in the EMR console (https://console.aws.amazon.com/elasticmapreduce/) or using the AWS CLI.
2. Set up a callback URL in your identity provider (IdP) account:
   - When using AD FS as your IdP, complete the following steps:
     1. From the AD FS Management Console, go to **Relying Party Trusts**.
     2. Right-click the display name of your replying party trust, and choose **Properties**.
     3. In the **Properties** window, choose the **Endpoints** tab.
     4. Select the temporary URL that you provided previously, then choose **edit**.
     5. In the **Edit Endpoint** window, update the Trusted URL with the correct DNS name for your master node.
     6. In the **Add an Endpoint** window, fill in the **Trusted URL** box with your master node public DNS. For example,

        ```
        https://ec2-11-111-11-111.compute-1.amazonaws.com:8442/gateway/knoxsso/api/v1/
        websso?pac4jCallback=true&client_name=SAML2Client
        ```

     7. Choose **OK**.
   - When using Auth0 as your IdP, complete the following steps:
     1. Go to https://auth0.com/ and log in.
     2. In the left panel, choose **Applications**.
     3. Select your previously created application.

4. On the **Settings** tab, update **Allowed Callback URLs** with your master node public DNS.

- When using Okta as your IdP, complete the following steps:
  1. Go to https://developer.okta.com/ and log in.
  2. In the top right corner, choose **Admin**, then choose the **Applications** tab.
  3. Select your application name.
  4. On the **General** tab under your application name, choose **SAML Settings**, then choose **Edit**.
  5. On the **Configure SAML** tab, update **Single-sign on URL** with your master node public DNS.

Set up EMR Notebooks to access Lake Formation data

After you opt in to data filtering on Amazon EMR and create a cluster integrated with Lake Formation, you can use both Apache Zeppelin and EMR Notebooks to access data.

In order to access both notebook applications, you must first ensure that your cluster's EC2 security group or VPC network access control list (ACL) is configured to allow access to the Proxy Agent (port 8442) from your desktop.

> **Note**
> The Proxy Agent on the EMR cluster uses a self-signed Transport Layer Security (TLS) certificate by default, and your browser will prompt you to accept the certificate before proceeding. If you want to use a custom certificate for the Proxy Agent, see the "Customize Proxy Agent Certificate" section.

Apache Zeppelin

To access Apache Zeppelin, use the EMR console to locate the *`Master public DNS`* from the cluster's **Summary** tab. Using your browser, navigate to https://*`MasterPublicDNS`*:8442/gateway/default/zeppelin/. Ensure the URL includes the trailing slash at the end.

Once the Proxy Agent's certificate is accepted, your browser redirects you to your Identity Provider (IdP) to authenticate. Once authenticated, you will be redirected to Zeppelin.

Creating your first Zeppelin Notebook

To get started, create a new Notebook by selecting **Notebook**, **Create new note**. Give your Notebook a name and use the default **livy** interpreter.

To see a list of Lake Formation databases, use the following Spark SQL command.

```
spark.sql("show databases").show()
```

To query a specific Lake Formation table, use the following Spark SQL command. Replace *`database.table`* with actual databases and tables in Lake Formation:

```
spark.sql("SELECT * FROM database.table limit 10").show()
```

EMR Notebooks

EMR Notebooks can be created using the Amazon EMR console and used with an existing EMR cluster integrated with Lake Formation.

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. Select **Choose an existing cluster**, and then **Choose**.
5. Select an existing EMR cluster integrated with Lake Formation.
6. Select **Create notebook** to create the notebook.

Once the Notebook has been created, select the notebook and click **Open**. You will be redirected to the Proxy Agent on the Amazon EMR cluster. Once you've accepted the Proxy Agent's certificate, your browser will redirect you to your Identity Provider (IdP) to authenticate. Once authenticated, you will be redirected to the EMR notebook.

For more information, see Using Amazon EMR Notebooks in the *Amazon EMR Management Guide*.

Customize your proxy agent certificate

The proxy agent uses a self-signed Transport Layer Security (TLS) certificate by default. You must replace the default certificate with a custom certificate for the proxy agent. To do that, you must first obtain a certificate, certificate chain, and private key from your certificate authority. With those items, a PKCS12 file will be used to protect the key material so it can be imported into the proxy agent's key store. The proxy agent is based on Apache Knox.

Take the following steps to replace the default certificate with your custom certificate. Replace *`MasterPublicDNS`* with 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`*.

1. To create a PKCS12 file from your certificate, certificate chain, and private key, run the following command on a host with the certificate files, and `openssl` installed.

   ```
   openssl pkcs12 -export -out proxy_agent_certificate.pfx -inkey private.key -
   in certificate.cer -certfile certchain.cer
   ```

2. Copy the `proxy_agent_certificate.pfx` file to the `/home/hadoop` directory on the master node of your cluster.

   ```
   scp -i EC2KeyPair.pem proxy_agent_certificate.pfx hadoop@MasterPublicDNS:/home/hadoop
   ```

3. SSH into the master node of your cluster.

   ```
   ssh -i EC2KeyPair.pem hadoop@MasterPublicDNS
   ```

4. Find your cluster-specific root key by using the following command.

   ```
   less /etc/knox/conf/gateway-site.xml
   ```

   Look for the *`gateway.master.secret`* property and copy the contents of the `value` tag, as you will need it for future steps.

5. Create a backup copy of the existing proxy agent keystores by using the following commands.

   ```
   sudo -s
   cd /mnt/var/lib/knox/data/security/keystores
   mkdir backups
   mv gateway.jks __gateway-credentials.jceks backups/
   ```

6. Import your custom certificates into a new keystore by using the following commands.

   ```
   sudo -s
   cd /mnt/var/lib/knox/data/security/keystores
   keytool -importkeystore \
   ```

```
-srckeystore /home/hadoop/proxy_agent_certificate.pfx \
-srcstoretype pkcs12 -destkeystore gateway.jks \
-deststoretype jks \
-srcalias 1 \
-destalias gateway-identity
```

When prompted for the `Enter destination keystore password`, use the Knox master secret from the `gateway-site.xml` file.

Ensure the newly created `gateway.jks` file is owned by the knox user by using the following command.

```
chown knox:knox gateway.jks
```

If your private key is protected by a password, make sure Knox is aware of that password.

```
sudo -u knox bash
cd /usr/lib/knox
bin/knoxcli.sh create-cert create-alias gateway-identity-passphrase
```

Enter the password protecting your private key when prompted.

7. Restart Knox by using the following commands.

```
sudo -u knox bash
cd /usr/lib/knox
bin/gateway.sh stop
```

Knox should be restarted automatically, and you can check the status of Knox by viewing the `/var/log/knox/gateway.log`.

8. To ensure the new certificate is being used by the proxy agent, navigate to Apache Zeppelin https://*MasterPublicDNS*:8442/gateway/default/zeppelin/. You can use your browser to inspect the certificate to ensure it is your custom certificate.

Set up cross-account access

After you launch an Amazon EMR cluster with Lake Formation integration, you can set up cross-account access. AWS Lake Formation cross-account access lets users query and join tables across multiple AWS accounts.

When you enable cross-account access, users from one account have granular access to Glue Data Catalog metadata and its underlying data that belongs to another account. For more information about cross-account access, see the *AWS Lake Formation Developer Guide*.

Overview

To set up cross-account access, you must complete tasks while logged in to the following AWS accounts:

- **Account A** - An AWS account with the Lake Formation data you want to share with an external account.
- **Account B** - The external AWS account to which you want to grant Lake Formation access. This should be an account where you have launched an EMR cluster with Lake Formation.

First, you set up a data resource in Account A and grant access to the resource for Account B. Then, you log in to Account B, accept the invitation from Account A, and create a resource link to give the resource

a new name in Account B. Finally, you add permissions to your Lake Formation IAM role from Account B so that users who are logged in to your cluster can make queries on the linked resource in Account A.

Prerequisites

To prepare your Data Catalog for cross-account access, follow the steps in Cross-account access prerequisites before you start. You should also read through Cross-account best practices and limitations to ensure the cross-account setup meets your data access requirements.

Set up cross-account access

To grant access to resources from Account A to Account B

1. Log in to the AWS Management Console with **Account A** and open the AWS Lake Formation console.
2. Follow the instructions in Requirements for roles used to register locations to create a *user-defined* service role for your data lake.
3. Register a storage location in Amazon S3 with the *user-defined role* you created in the previous step using the instructions in Registering an Amazon S3 location.

   Warning
   You must use a user-defined role and not the Lake Formation service-linked role when you register this data location. Lake Formation does not support using its service-linked role when you integrate with EMR.
4. Create a database using the data location you just registered. For instructions, see Creating a database.
5. Create a table in the database you set up in the previous step. For instructions, see Creating tables.
6. Grant permissions to **Account B**. Specify the following values:
   - The database you created in step 3.
   - The table you created in step 4.
   - The account ID of the external AWS account to which you want to grant access (**Account B**).
   - Check the box next to **Select** under both **Table** and **Grantable** permissions. You must also select the **Describe** permission for the table. When you grant permissions to an external account, the permissions do not extend to any of the principals in that external account. You must allow grantable permissions so that Account B can *in turn* grant access to its IAM role for Lake Formation principal.
7. Complete the steps in Allow data filtering on Amazon EMR in the *Lake Formation Developer Guide* to opt in to data filtering for data lakes on Amazon EMR in Account A. Specify the ID for Account B under **AWS account IDs** on the **External data filtering** page.

To finish setting up cross-account access in Account B

1. After you complete the preceding steps in Account A, log in to the AWS Management Console with **Account B**.
2. In the AWS Resource Access Manager console, accept the shared resource invitation from Account A. For instructions, see Accessing resources shared with you in the *AWS Resource Access Manager User Guide*.
3. Access the AWS Lake Formation console as a Data lake administrator.
4. Create a database resource link for the remote database shared by Account A. Creating a resource link lets you assign a different name to the remote database in your Data Catalog. For instructions, see Creating a resource link to a shared data catalog database.
5. On the **Databases** page, choose the database resource link you created previously.
6. In the **Actions** dropdown under **Permissions**, select **Grant**.

7. For **SAML and Amazon QuickSight users and groups**, select the SAML user(s) or group(s) you want to give access to, and make sure **Describe** is enabled under **Resource Link Permissions**. For more information, see Granting resource link permissions.
8. Choose **Grant** to finish granting permissions to your resource link.
9. In the **Actions** dropdown under **Permissions**, select **Grant on target**. This step is required since granting permissions on a resource link does not grant permissions on a target (linked) database or table. You must grant permissions on a target separately. For more information, see Granting resource link permissions.
10. For **SAML and Amazon QuickSight users and groups**, select your the same SAML user(s) or group(s) you chose in Step 7.
11. Choose appropriate target permissions for your IAM role for Lake Formation, then choose **Grant**.

You should now be able to access a resource in Account A from your EMR cluster in Account B.

For example, say you created a resource link called **resource-db-link** and granted access to all the columns of a table, **t1**, in the **resource-db-link** database. You can now retrieve the first column, **col1**, with the following Spark SQL query:

```
select col1 from resource-db-link.t1
```

For information about troubleshooting cross-account access, see Troubleshooting Lake Formation in the *AWS Lake Formation Developer Guide*.

Integrate Amazon EMR with Apache Ranger

Beginning with Amazon EMR 5.32.0, you can launch a cluster that natively integrates with Apache Ranger. Apache Ranger is an open-source framework to enable, monitor, and manage comprehensive data security across the Hadoop platform. For more information, see Apache Ranger. With native integration, you can bring your own Apache Ranger to enforce fine-grained data access control on Amazon EMR.

This section provides a conceptual overview of Amazon EMR integration with Apache Ranger. It also includes the prerequisites and steps required to launch an Amazon EMR cluster integrated with Apache Ranger.

Natively integrating Amazon EMR with Apache Ranger provides the following key benefits:

- Fine-grained access control to Hive Metastore databases and tables, which enables you to define data filtering policies at the level of database, table, and column for Apache Spark and Apache Hive applications. Row-level filtering and data masking are supported with Hive applications.
- The ability to use your existing Hive policies directly with Amazon EMR for Hive applications.
- Access control to Amazon S3 data at the prefix and object level, which enables you to define data filtering policies for access to S3 data using the EMR File System.
- The ability to use CloudWatch Logs for centralized auditing.
- Amazon EMR installs and manages the Apache Ranger plugins on your behalf.

Apache Ranger

Apache Ranger is a framework to enable, monitor, and manage comprehensive data security across the Hadoop platform.

Apache Ranger has the following features:

- Centralized security administration to manage all security related tasks in a central UI or using REST APIs.
- Fine-grained authorization to do a specific action or operation with a Hadoop component or tool, managed through a central administration tool.
- A standardized authorization method across all Hadoop components.
- Enhanced support for various authorization methods.
- Centralized auditing of user access and administrative actions (security related) within all the components of Hadoop.

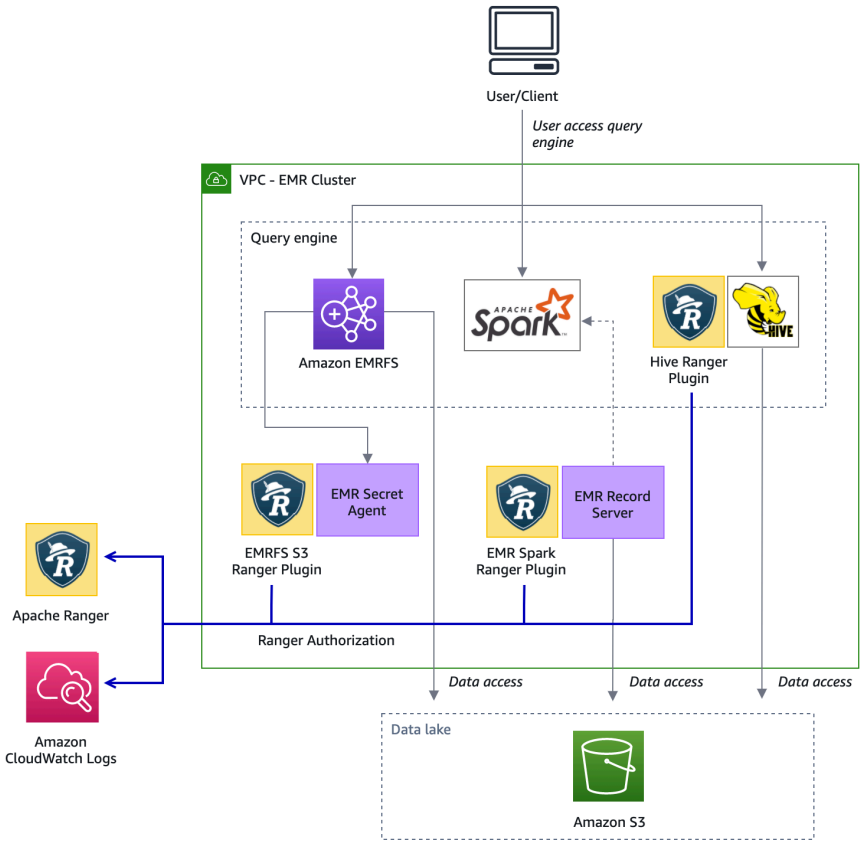
Apache Ranger uses two key components for authorization:

- **Apache Ranger policy admin server** - This server allows you to define the authorization policies for Hadoop applications. When integrating with Amazon EMR, you are able to define and enforce policies for Apache Spark and Hive to access Hive Metastore, and accessing Amazon S3 data EMR File System (EMRFS). You can set up a new or use an existing Apache Ranger policy admin server to integrate with Amazon EMR.
- **Apache Ranger plugin** - This plugin validates the access of a user against the authorization policies defined in the Apache Ranger policy admin server. Amazon EMR installs and configures the Apache Ranger plugin automatically for each Hadoop application selected in the Apache Ranger configuration.

Topics

- Architecture of Amazon EMR integration with Apache Ranger (p. 497)
- Amazon EMR components (p. 497)

Architecture of Amazon EMR integration with Apache Ranger



Amazon EMR components

Amazon EMR enables fine-grained access control with Apache Ranger through the following components. See the architecture diagram (p. 497) for a visual representation of these Amazon EMR components with the Apache Ranger plugins.

Secret agent – The secret agent securely stores secrets and distributes secrets to other Amazon EMR components or applications. The secrets can include temporary user credentials, encryption keys, or Kerberos tickets. The secret agent runs on every node in the cluster and intercepts calls to the Instance Metadata Service. For requests to the instance profile role credentials, the Secret Agent vends credentials depending on the requesting user and requested resources after authorizing the request with the EMRFS S3 Ranger plugin. The secret agent runs as the *`emrsecretagent`* user, and it writes logs to the /emr/secretagent/log directory. The process relies on a specific set of `iptables` rules to function. It is important to ensure that `iptables` is not disabled. If you customize `iptables` configuration, the NAT table rules must be preserved and left unaltered.

EMR record server – The record server receives requests to access data from Spark. It then authorizes requests by forwarding the requested resources to the Spark Ranger plugin for Amazon EMR. The record server reads data from Amazon S3 and returns filtered data that the user is authorized to access based on Ranger policy. The record server runs on every node in the cluster as the emr_record_server user and writes logs to the /var/log/emr-record-server directory.

Application support and limitations

Supported applications

The integration between Amazon EMR and Apache Ranger in which EMR installs Ranger plugins currently supports the following applications:

- Apache Spark (Available with EMR 5.32+ and EMR 6.3+)
- Apache Hive (Available with EMR 5.32+ and EMR 6.3+)
- S3 Access through EMRFS (Available with EMR 5.32+ and EMR 6.3+)

The following applications can be installed on an EMR cluster and may need to be configured to meet your security needs:

- Apache Hadoop (Available with EMR 5.32+ and EMR 6.3+ including YARN and HDFS)
- Apache Livy (Available with EMR 5.32+ and EMR 6.3+)
- Apache Zeppelin (Available with EMR 5.32+ and EMR 6.3+)
- Apache Hue (Available with EMR 5.32+ and EMR 6.3+)
- Ganglia (Available with EMR 5.32+ and EMR 6.3+)
- HCatalog (Available with EMR 5.32+ and EMR 6.3+)
- Mahout (Available with EMR 5.32+ and EMR 6.3+)
- MXNet (Available with EMR 5.32+ and EMR 6.3+)
- TensorFlow (Available with EMR 5.32+ and EMR 6.3+)
- Tez (Available with EMR 5.32+ and EMR 6.3+)
- Trino (Available with EMR 6.7+)
- ZooKeeper (Available with EMR 5.32+ and EMR 6.3+)

> **Important**
> Applications listed above are the only applications that are currently supported. To ensure cluster security, you are allowed to create an EMR cluster with only the applications in the above list when Apache Ranger is enabled.
> Other applications are currently not supported. To ensure the security of your cluster, attempting to install other applications will cause the rejection your cluster.

Supported Features

The following Amazon EMR features can be used with Amazon EMR and Apache Ranger:

- Encryption at rest and in transit
- Kerberos authentication (required)
- Instance groups, instance fleets, and Spot Instances
- Reconfiguration of applications on a running cluster
- EMRFS server-side encryption (SSE)

> **Note**
> Amazon EMR encryption settings govern SSE. For more information, see Encryption Options (p. 374).

Application limitations

There are several limitations to keep in mind when you integrate Amazon EMR and Apache Ranger:

- You cannot currently use the console to create a security configuration that specifies the AWS Ranger integration option in the AWS GovCloud (US) Region. Security configuration can be done using the CLI.
- Kerberos must be installed on your cluster.
- Application UIs (user interfaces) such as the YARN Resource Manager UI, HDFS NameNode UI, and Livy UI are not set with authentication by default.
- The HDFS default permissions `umask` are configured so that objects created are set to `world wide readable` by default.
- Amazon EMR doesn't support high-availability (multiple primary) mode with Apache Ranger.
- For additional limitations, see limitations for each application.

> **Note**
> Amazon EMR encryption settings govern SSE. For more information, see Encryption Options (p. 374).

Plugin limitations

Each plugin has specific limitations. For the Apache Hive plugin's limitations, see Apache Hive plugin limitations. For the Apache Spark plugin's limitations, see Apache Spark plugin limitations. For the EMRFS S3 plugin's limitations, see EMRFS S3 plugin limitations.

Set up Amazon EMR for Apache Ranger

Before you install Apache Ranger, review the information in this section to make sure that Amazon EMR is properly configured.

Topics

- Set up Ranger Admin server (p. 499)
- IAM roles for native integration with Apache Ranger (p. 502)
- Create the EMR security configuration (p. 504)
- Store TLS certificates in AWS Secrets Manager (p. 506)
- Start an EMR cluster (p. 508)
- Configure Zeppelin for Apache Ranger-enabled Amazon EMR clusters (p. 508)
- Known issues (p. 510)

Set up Ranger Admin server

For Amazon EMR integration, the Apache Ranger application plugins must communicate with the Admin server using TLS/SSL.

Prerequisite: Ranger Admin Server SSL Enablement

Apache Ranger on Amazon EMR requires two-way SSL communication between plugins and the Ranger Admin server. To ensure that plugins communicate with the Apache Ranger server over SSL, enable the following attribute within ranger-admin-site.xml on the Ranger Admin server.

```
<property>
    <name>ranger.service.https.attrib.ssl.enabled</name>
    <value>true</value>
</property>
```

In addition, the following configurations are needed.

```
<property>
    <name>ranger.https.attrib.keystore.file</name>
    <value>_<PATH_TO_KEYSTORE>_</value>
</property>

<property>
    <name>ranger.service.https.attrib.keystore.file</name>
    <value>_<PATH_TO_KEYSTORE>_</value>
</property>

<property>
    <name>ranger.service.https.attrib.keystore.pass</name>
    <value>_<KEYSTORE_PASSWORD>_</value>
</property>

<property>
    <name>ranger.service.https.attrib.keystore.keyalias</name>
    <value><PRIVATE_CERTIFICATE_KEY_ALIAS></value>
</property>

<property>
    <name>ranger.service.https.attrib.clientAuth</name>
    <value>want</value>
</property>

<property>
    <name>ranger.service.https.port</name>
    <value>6182</value>
</property>
```

TLS certificates

Apache Ranger integration with Amazon EMR requires that traffic from Amazon EMR nodes to the Ranger Admin server is encrypted using TLS, and that Ranger plugins authenticate to the Apache Ranger server using two-way mutual TLS authentication. Amazon EMR service needs the public certificate of your Ranger Admin server (specified in the previous example) and the private certificate.

Apache Ranger plugin certificates

Apache Ranger plugin public TLS certificates must be accessible to the Apache Ranger Admin server to validate when the plugins connect. There are three different methods to do this.

Method 1: Configure a truststore in Apache Ranger Admin server

Fill in the following configurations in ranger-admin-site.xml to configure a truststore.

```
<property>
    <name>ranger.truststore.file</name>
    <value><LOCATION TO TRUSTSTORE></value>
</property>

<property>
    <name>ranger.truststore.password</name>
    <value><PASSWORD FOR TRUSTSTORE></value>
</property>
```

Method 2: Load the certificate into Java cacerts truststore

If your Ranger Admin server doesn't specify a truststore in its JVM options, then you can put the plugin public certificates in the default cacerts store.

Method 3: Create a truststore and specify as part of JVM Options

Within `{RANGER_HOME_DIRECTORY}/ews/ranger-admin-services.sh`, modify `JAVA_OPTS` to include "`-Djavax.net.ssl.trustStore=<TRUSTSTORE_LOCATION>`" and "`-Djavax.net.ssl.trustStorePassword=<TRUSTSTORE_PASSWORD>`". For example, add the following line after the existing JAVA_OPTS.

```
JAVA_OPTS=" ${JAVA_OPTS} -Djavax.net.ssl.trustStore=${RANGER_HOME}/truststore/
truststore.jck -Djavax.net.ssl.trustStorePassword=changeit"
```

> **Note**
> This specification may expose the truststore password if any user is able to log into the Apache Ranger Admin server and see running processes, such as when using the `ps` command.

Using Self-Signed Certificates

Self-signed certificates are not recommended as certificates. Self-signed certificates may not be revoked, and self-signed certificates may not conform to internal security requirements.

Service definition installation

A service definition is used by the Ranger Admin server to describe the attributes of policies for an application. The policies are then stored in a policy repository for clients to download.

To be able to configure service definitions, REST calls must be made to the Ranger Admin server. See Apache Ranger PublicAPIsv2for APIs required in the following section.

Installing Apache Spark's Service Definition

To install Apache Spark's service definition, see Apache Spark plugin (p. 514).

Installing EMRFS Service Definition

To install the S3 service definition for Amazon EMR, see EMRFS S3 plugin (p. 519).

Using Hive Service Definition

Apache Hive can use the existing Ranger service definition that ships with Apache Ranger 2.0 and later. For more information, see Apache Hive plugin (p. 511).

Network traffic rules

When Apache Ranger is integrated with your EMR cluster, the cluster needs to communicate with additional servers and AWS.

All Amazon EMR nodes, including core and task nodes, must be able to communicate with the Apache Ranger Admin servers to download policies. If your Apache Ranger Admin is running on Amazon EC2, you need to update the security group to be able to take traffic from the EMR cluster.

In addition to communicating with the Ranger Admin server, all nodes need to be able to communicate with the following AWS services:

- Amazon S3
- AWS KMS (if using EMRFS SSE-KMS)
- Amazon CloudWatch
- AWS STS

If you are planning to run your EMR cluster within a private subnet, configure the VPC to be able to communicate with these services using either AWS PrivateLink and VPC endpoints in the *Amazon VPC User Guide* or using network address translation (NAT) instance in the *Amazon VPC User Guide*.

IAM roles for native integration with Apache Ranger

The integration between Amazon EMR and Apache Ranger relies on three key roles that you should create before you launch your cluster:

- A custom Amazon EC2 instance profile for Amazon EMR
- An IAM role for Apache Ranger Engines
- An IAM role for other AWS services

This section gives an overview of these roles and the policies that you need to include for each IAM role. For information about creating these roles, see Set up Ranger Admin server (p. 499).

EC2 instance profile

Amazon EMR uses an IAM service roles to perform actions on your behalf to provision and manage clusters. The service role for cluster EC2 instances, also called the EC2 instance profile for Amazon EMR, is a special type of service role assigned to every EC2 instance in a cluster at launch.

To define permissions for EMR cluster interaction with Amazon S3 data and with Hive metastore protected by Apache Ranger and other AWS services, define a custom EC2 instance profile to use instead of the EMR_EC2_DefaultRole when you launch your cluster.

For more information, see Service role for cluster EC2 instances (EC2 instance profile) (p. 403) and Customize IAM roles (p. 418).

You need to add the following statements to the default EC2 Instance Profile for Amazon EMR to be able to tag sessions and access the AWS Secrets Manager that stores TLS certificates.

```
    {
      "Sid": "AllowAssumeOfRolesAndTagging",
      "Effect": "Allow",
      "Action": ["sts:TagSession", "sts:AssumeRole"],
      "Resource": [
        "arn:aws:iam::<AWS_ACCOUNT_ID>:role/<RANGER_ENGINE-PLUGIN_DATA_ACCESS_ROLE_NAME>",
        "arn:aws:iam::<AWS_ACCOUNT_ID>:role/<RANGER_USER_ACCESS_ROLE_NAME>"
      ]
    },
    {
        "Sid": "AllowSecretsRetrieval",
        "Effect": "Allow",
        "Action": "secretsmanager:GetSecretValue",
        "Resource": [

 "arn:aws:secretsmanager:<REGION>:<AWS_ACCOUNT_ID>:secret:<PLUGIN_TLS_SECRET_NAME>*",

 "arn:aws:secretsmanager:<REGION>:<AWS_ACCOUNT_ID>:secret:<ADMIN_RANGER_SERVER_TLS_SECRET_NAME>*"
        ]
    }
```

Note
For the Secrets Manager permissions, do not forget the wildcard ("*") at the end of the secret name or your requests will fail. The wildcard is for secret versions.

Note
Limit the scope of the AWS Secrets Manager policy to only the certificates that are required for provisioning.

IAM role for Apache Ranger

This role provides credentials for trusted execution engines, such as Apache Hive and Amazon EMR Record Server Amazon EMR components (p. 477), to access Amazon S3 data. Use only this role to access Amazon S3 data, including any KMS keys, if you are using S3 SSE-KMS.

This role must be created with the minimum policy stated in the following example.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CloudwatchLogsPermissions",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Effect": "Allow",
      "Resource": [

 "arn:aws:logs:<REGION>:<AWS_ACCOUNT_ID>:<CLOUDWATCH_LOG_GROUP_NAME_IN_SECURITY_CONFIGURATION>:*"
      ]
    },
    {
      "Sid": "BucketPermissionsInS3Buckets",
      "Action": [
        "s3:CreateBucket",
        "s3:DeleteBucket",
        "s3:ListAllMyBuckets",
        "s3:ListBucket"
      ],
      "Effect": "Allow",
      "Resource": [
 *"arn:aws:s3:::bucket1",
        "arn:aws:s3:::bucket2"*
        ]
    },
    {
      "Sid": "ObjectPermissionsInS3Objects",
      "Action": [
        "s3:GetObject",
        "s3:DeleteObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": [
 *"arn:aws:s3:::bucket1/*",
          "arn:aws:s3:::bucket2/*"
*         ]
    }
  ]
}
```

Important
The asterisk "*" at the end of the CloudWatch Log Resource must be included to provide permission to write to the log streams.

Note
If you are using EMRFS consistency view or S3-SSE encryption, add permissions to the DynamoDB tables and KMS keys so that execution engines can interact with those engines.

The IAM role for Apache Ranger is assumed by the EC2 Instance Profile Role. Use the following example to create a trust policy that allows the IAM role for Apache Ranger to be assumed by the EC2 instance profile role.

```
{
  "Sid": "",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::<AWS_ACCOUNT_ID>:role/<EC2 INSTANCE PROFILE ROLE NAME eg.
EMR_EC2_DefaultRole>"
  },
  "Action": ["sts:AssumeRole", "sts:TagSession"]
}
```

IAM role for other AWS services

This role provides users who are not trusted execution engines with credentials to interact with AWS services, if needed. Do not use this IAM role to allow access to Amazon S3 data, unless it's data that should be accessible by all users.

This role will be assumed by the EC2 Instance Profile Role. Use the following example to create a trust policy that allows the IAM role for Apache Ranger to be assumed by the EC2 instance profile role.

```
{
  "Sid": "",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::<AWS_ACCOUNT_ID>:role/<EC2 INSTANCE PROFILE ROLE NAME eg.
EMR_EC2_DefaultRole>"
  },
  "Action": ["sts:AssumeRole", "sts:TagSession"]
}
```

Validate your permissions

See Apache Ranger troubleshooting (p. 530) for instructions on validating permissions.

Create the EMR security configuration

Creating an Amazon EMR Security Configuration for Apache Ranger

Before you launch an Amazon EMR cluster integrated with Apache Ranger, create a security configuration.

Console

To create a security configuration that specifies the AWS Ranger integration option

1. In the Amazon EMR console, select **Security configurations**, then **Create**.
2. Type a **Name** for the security configuration. You use this name to specify the security configuration when you create a cluster.
3. Under **AWS Ranger Integration**, select **Enable fine-grained access control managed by Apache Ranger**.
4. Select your **IAM role for Apache Ranger** to apply. For more information, see IAM roles for native integration with Apache Ranger (p. 502).
5. Select your **IAM role for other AWS services** to apply.
6. Configure the plugins to connect to the Ranger Admin server by entering the Secret Manager ARN for the Admin server and the address.

7. Select the applications to configure Ranger plugins. Fill in the Secret Manager ARN that contain the private TLS certificate for the plugin.

   If you do not configure Apache Spark or Apache Hive, and they are selected as an application for your cluster, the request fails.
8. Set up other security configuration options as appropriate and choose **Create**. You must enable Kerberos authentication using the cluster-dedicated or external KDC. For more information, see Configure EMR security (p. 487).

   Note
   You cannot currently use the console to create a security configuration that specifies the AWS Ranger integration option in the AWS GovCloud (US) Region. Security configuration can be done using the CLI.

CLI

To create a security configuration for Apache Ranger integration

1. Replace *<ACCOUNT ID>* with your AWS account ID.
2. Replace *<REGION>* with the Region that the resource is in.
3. Specify a value for `TicketLifetimeInHours` to determine the period for which a Kerberos ticket issued by the KDC is valid.
4. Specify the address of the Ranger Admin server for `AdminServerURL`.

```
{
    "AuthenticationConfiguration": {
        "KerberosConfiguration": {
            "Provider": "ClusterDedicatedKdc",
            "ClusterDedicatedKdcConfiguration": {
                "TicketLifetimeInHours": 24
            }
        }
    },
    "AuthorizationConfiguration":{
      "RangerConfiguration":{
         "AdminServerURL":"https://_<RANGER ADMIN SERVER IP>_:6182",
         "RoleForRangerPluginsARN":"arn:aws:iam::_<ACCOUNT ID>_:role/_<RANGER PLUGIN
 DATA ACCESS ROLE NAME>_",
         "RoleForOtherAWSServicesARN":"arn:aws:iam::_<ACCOUNT ID>_:role/_<USER ACCESS
 ROLE NAME>_",
         "AdminServerSecretARN":"arn:aws:secretsmanager:_<REGION>_:_<ACCOUNT
 ID>_:secret:_<SECRET NAME THAT PROVIDES ADMIN SERVERS PUBLIC TLS CERTICATE WITHOUT
 VERSION>_",
         "RangerPluginConfigurations":[
            {
               "App":"Spark",
               "ClientSecretARN":"arn:aws:secretsmanager:_<REGION>_:_<ACCOUNT
 ID>_:secret:_<SECRET NAME THAT PROVIDES SPARK PLUGIN PRIVATE TLS CERTICATE WITHOUT
 VERSION>_",
               "PolicyRepositoryName":"<SPARK SERVICE NAME eg. amazon-emr-spark>"
            },
            {
               "App":"Hive",
               "ClientSecretARN":"arn:aws:secretsmanager:_<REGION>_:_<ACCOUNT
 ID>_:secret:_<SECRET NAME THAT PROVIDES Hive PLUGIN PRIVATE TLS CERTICATE WITHOUT
 VERSION>_",
               "PolicyRepositoryName":"<HIVE SERVICE NAME eg. Hivedev>"
            },
            {
               "App":"EMRFS-S3",
```

```
                "ClientSecretARN":"arn:aws:secretsmanager:_<REGION>_:_<ACCOUNT
 ID>_:secret:_<SECRET NAME THAT PROVIDES EMRFS S3 PLUGIN PRIVATE TLS CERTICATE WITHOUT
 VERSION>_",
                "PolicyRepositoryName":"<EMRFS S3 SERVICE NAME eg amazon-emr-emrfs>"
            }
         ],
         "AuditConfiguration":{
            "Destinations":{
               "AmazonCloudWatchLogs":{
                  "CloudWatchLogGroup":"arn:aws:logs:<REGION>:_<ACCOUNT ID>_:log-
group:_<LOG GROUP NAME FOR AUDIT EVENTS>_"
               }
            }
         }
      }
   }
}
```

The PolicyRespositoryNames are the service names that are specified in your Apache Ranger Admin.

Create an Amazon EMR security configuration with the following command. Replace security-configuration with a name of your choice. Select this configuration by name when you create your cluster.

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

Configure Additional Security Features

To securely integrate Amazon EMR with Apache Ranger, configure the following EMR security features:

- Enable Kerberos authentication using the cluster-dedicated or external KDC. For instructions, see Use Kerberos authentication (p. 448).
- (Optional) Enable encryption in transit or at rest. For more information, see Encryption options (p. 374).

For more information, see Security in Amazon EMR (p. 353).

Store TLS certificates in AWS Secrets Manager

The Ranger plugins installed on an Amazon EMR cluster and the Ranger Admin server must communicate over TLS to ensure that policy data and other information sent cannot be read if they are intercepted. EMR also mandates that the plugins authenticate to the Ranger Admin server by providing its own TLS certificate and perform two-way TLS authentication. This setup required four certificates to be created: two pairs of private and public TLS certificates. For instructions on installing the certificate to your Ranger Admin server, see Set up Ranger Admin server (p. 499). To complete the setup, the Ranger plugins installed on the EMR cluster need two certificates: the public TLS certificate of your admin server, and the private certificate that the plugin will use to authenticate against the Ranger Admin server. To provide these TLS certificates, they must be in the AWS Secrets Manager and provided in a EMR Security Configuration.

> **Note**
> It is strongly recommended, but not required, to create a certificate pair for each of your applications to limit impact if one of the plugin certificates becomes compromised.

> **Note**
> You need to track and rotate certificates prior to their expiration date.

Certificate format

Importing the certificates to the AWS Secrets Manager is the same regardless of whether it is the private plugin certificate or the public Ranger admin certificate. Before importing the TLS certificates, the certificates must be in 509x PEM format.

An example of a public certificate is in the format:

```
-----BEGIN CERTIFICATE-----
...Certifcate Body...
-----END CERTIFICATE-----
```

An example of a private certificate is in the format:

```
-----BEGIN PRIVATE KEY-----
...Private Certifcate Body...
-----END PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
...Trust Certificate Body...
-----END CERTIFICATE-----
```

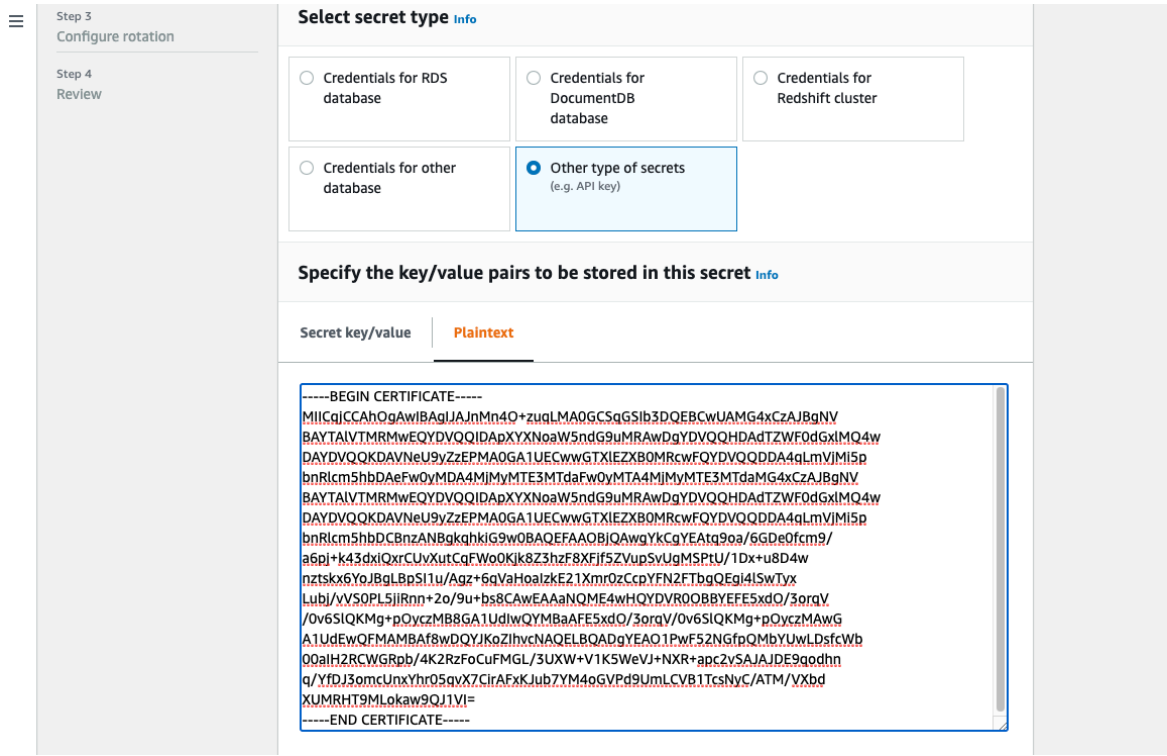
The private certificate should also contain a trust certificate as well.

You can validate that the certificates are in the correct format by running the following command:

```
openssl x509 -in <PEM FILE> -text
```

Importing a certificate to the AWS Secrets Manager

When creating your Secret in the Secrets Manager, choose **Other type of secrets** under **secret type** and paste your PEM encoded certificate in the **Plaintext** field.



Start an EMR cluster

Before you launch an Amazon EMR cluster with Apache Ranger, make sure each component meets the following minimum version requirement:

- Amazon EMR 5.32.0 or later, or 6.3.0 or later. We recommend that you use the latest Amazon EMR release version.
- Apache Ranger Admin server 2.x.

Complete the following steps.

- Install Apache Ranger if you haven't already. For more information, see Apache Ranger 0.5.0 installation.
- Make sure there is network connectivity between your Amazon EMR cluster and the Apache Ranger Admin server. See Set up Ranger Admin server (p. 499)
- Create the necessary IAM Roles. See IAM roles for native integration with Apache Ranger (p. 502).
- Create a EMR security configuration for Apache Ranger installation. See more information, see Create the EMR security configuration (p. 504).

Configure Zeppelin for Apache Ranger-enabled Amazon EMR clusters

The topic covers how to configure Apache Zeppelin for an Apache Ranger-enabled Amazon EMR cluster so that you can use Zeppelin as a notebook for interactive data exploration. Zeppelin is included in Amazon EMR release versions 5.0.0 and later. Earlier release versions include Zeppelin as a sandbox application. For more information, see Amazon EMR 4.x release versions in the *Amazon EMR Release Guide*.

By default, Zeppelin is configured with a default login and password which is not secure in a multi-tenant environment.

To configure Zeppelin, complete the following steps.

1. **Modify the authentication mechanism**.

   Modify the `shiro.ini` file to implement your preferred authentication mechanism. Zeppelin supports Active Directory, LDAP, PAM, and Knox SSO. See Apache Shiro authentication for Apache Zeppelin for more information.

2. **Configure Zeppelin to impersonate the end user**

   When you allow Zeppelin to impersonate the end user, jobs submitted by Zeppelin can be run as that end user. Add the following configuration to `core-site.xml`:

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

Next, add the following configuration to `hadoop-kms-site.xml` located in `/etc/hadoop/conf`:

```
[
  {
    "Classification": "hadoop-kms-site",
    "Properties": {
      "hadoop.kms.proxyuser.zeppelin.hosts": "*",
      "hadoop.kms.proxyuser.zeppelin.groups": "*"
    },
    "Configurations": [
    ]
  }
]
```

You can also add these configurations to your Amazon EMR cluster using the console by following the steps in Reconfigure an instance group in the console.

3. **Allow Zeppelin to sudo as the end user**

   Create a file `/etc/sudoers.d/90-zeppelin-user` that contains the following:

```
zeppelin ALL=(ALL) NOPASSWD:ALL
```

4. **Modify interpreters settings to run user jobs in their own processes.**

   For all interpreters, configure them to instantiate the interpreters "Per User" in "isolated" processes.

5. **Modify `zeppelin-env.sh`**

   Add the following to `zeppelin-env.sh` so that Zeppelin starts launch interpreters as the end user:

```
ZEPPELIN_IMPERSONATE_USER=`echo ${ZEPPELIN_IMPERSONATE_USER} | cut -d @ -f1`
export ZEPPELIN_IMPERSONATE_CMD='sudo -H -u ${ZEPPELIN_IMPERSONATE_USER} bash -c'
```

   Add the following to `zeppelin-env.sh` to change the default notebook permissions to read-only to the creator only:

```
export ZEPPELIN_NOTEBOOK_PUBLIC="false"
```

   Finally, add the following to `zeppelin-env.sh` to include the EMR RecordServer class path after the first `CLASSPATH` statement:

```
export CLASSPATH="$CLASSPATH:/usr/share/aws/emr/record-server/lib/aws-emr-record-server-connector-common.jar:/usr/share/aws/emr/record-server/lib/aws-emr-record-server-spark-connector.jar:/usr/share/aws/emr/record-server/lib/aws-emr-record-server-client.jar:/usr/share/aws/emr/record-server/lib/aws-emr-record-server-common.jar:/usr/share/aws/emr/record-server/lib/jars/secret-agent-interface.jar"
```

6. **Restart Zeppelin.**

   Run the following command to restart Zeppelin:

```
sudo systemctl restart zeppelin
```

Known issues

Known Issues

There is a known issue within Amazon EMR release 5.32 in which the permissions for `hive-site.xml` was changed so that only privileged users can read it as there may be credentials stored within it. This could prevent Hue from reading `hive-site.xml` and cause webpages to continuously reload. If you experience this issue, add the following configuration to fix the issue:

```
[
  {
    "Classification": "hue-ini",
    "Properties": {},
    "Configurations": [
      {
        "Classification": "desktop",
        "Properties": {
          "server_group":"hive_site_reader"
         },
        "Configurations":[
        ]
      }
    ]
  }
]
```

There is a known issue that the EMRFS S3 plugin for Apache Ranger currently does not support Apache Ranger's Security Zone feature. Access control restrictions defined using the Security Zone feature are not applied on your Amazon EMR clusters.

Application UIs

By default, Application UI's do not perform authentication. This includes the ResourceManager UI, NodeManager UI, Livy UI, among others. In addition, any user that has the ability to access the UIs is able to view information about all other users' jobs.

If this behavior is not desired, you should ensure that a security group is used to restrict access to the application UIs by users.

HDFS Default Permissions

By default, the objects that users create in HDFS are given world readable permissions. This can potentially cause data readable by users that should not have access to it. To change this behavior such that the default file permissions are set to read and write only by the creator of the job, perform these steps.

When creating your EMR cluster, provide the following configuration:

```
[
  {
    "Classification": "hdfs-site",
    "Properties": {
      "dfs.namenode.acls.enabled": "true",
      "fs.permissions.umask-mode": "077",
```

```
      "dfs.permissions.superusergroup": "hdfsadmingroup"
    }
  }
]
```

In addition, run the following bootstrap action:

```
--bootstrap-actions Name='HDFS UMask Setup',Path=s3://elasticmapreduce/hdfs/umask/umask-
main.sh
```

Apache Ranger plugins

Apache Ranger plugins validate the access of a user against the authorization policies defined in the Apache Ranger policy admin server.

Topics

- Apache Hive plugin (p. 511)
- Apache Spark plugin (p. 514)
- EMRFS S3 plugin (p. 519)
- Trino plugin (p. 525)

Apache Hive plugin

Apache Hive is a popular execution engine within the Hadoop ecosystem. Amazon EMR provides an Apache Ranger plugin to be able to provide fine-grained access controls for Hive. The plugin is compatible with open source Apache Ranger Admin server version 2.0 and later.

Topics

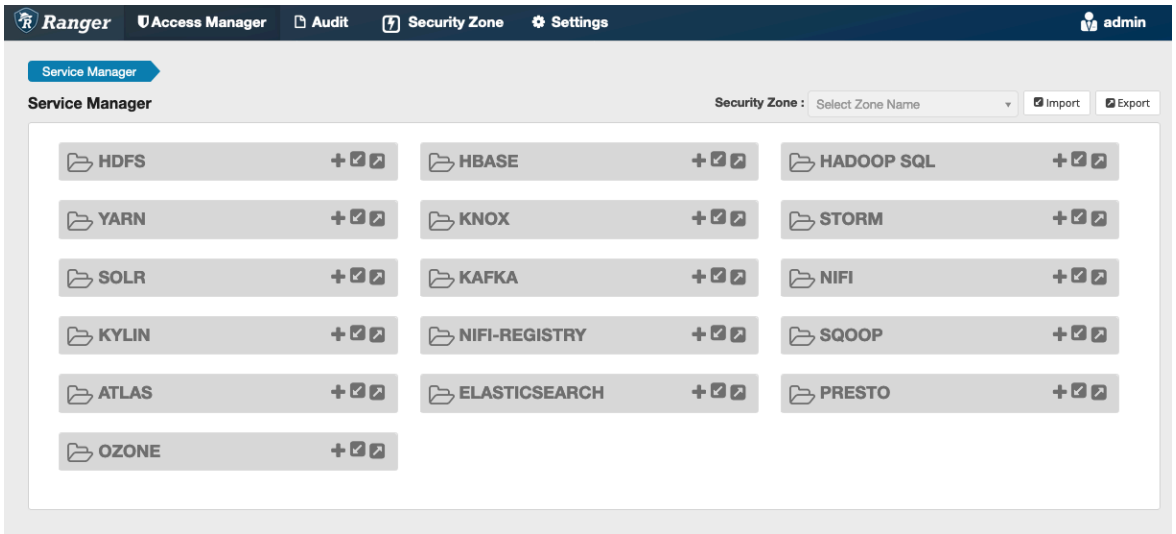
- Supported features (p. 511)
- Installation of service configuration (p. 511)
- Considerations (p. 513)
- Limitations (p. 514)

Supported features

The Apache Ranger plugin for Hive on EMR supports all the functionality of the open source plugin, which includes database, table, column level access controls and row filtering and data masking. For a table of Hive commands and associated Ranger permissions, see Hive commands to Ranger permission mapping.

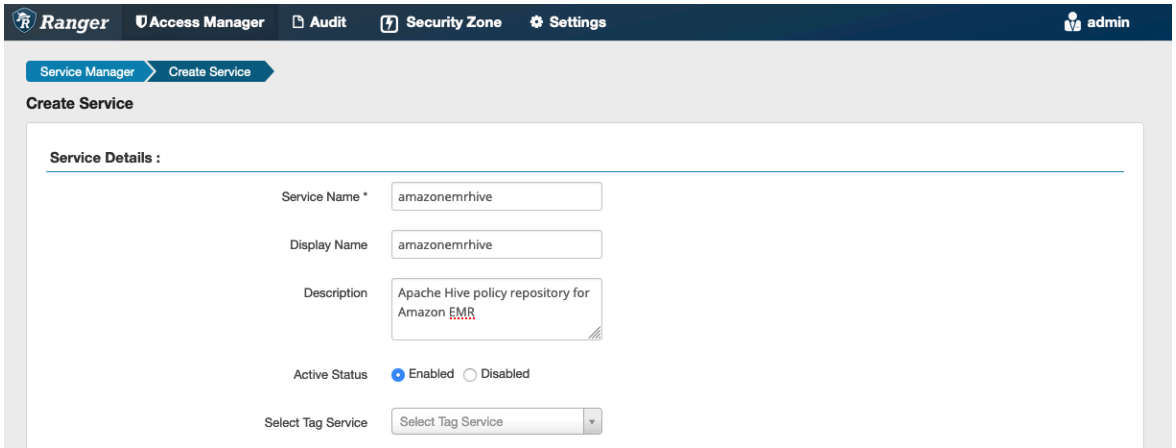
Installation of service configuration

The Apache Hive plugin is compatible with the existing Hive service definition within Apache Hive Hadoop SQL.



If you do not have an instance of the service under Hadoop SQL, like shown above, you can create one. Click on the **+** next to Hadoop SQL.

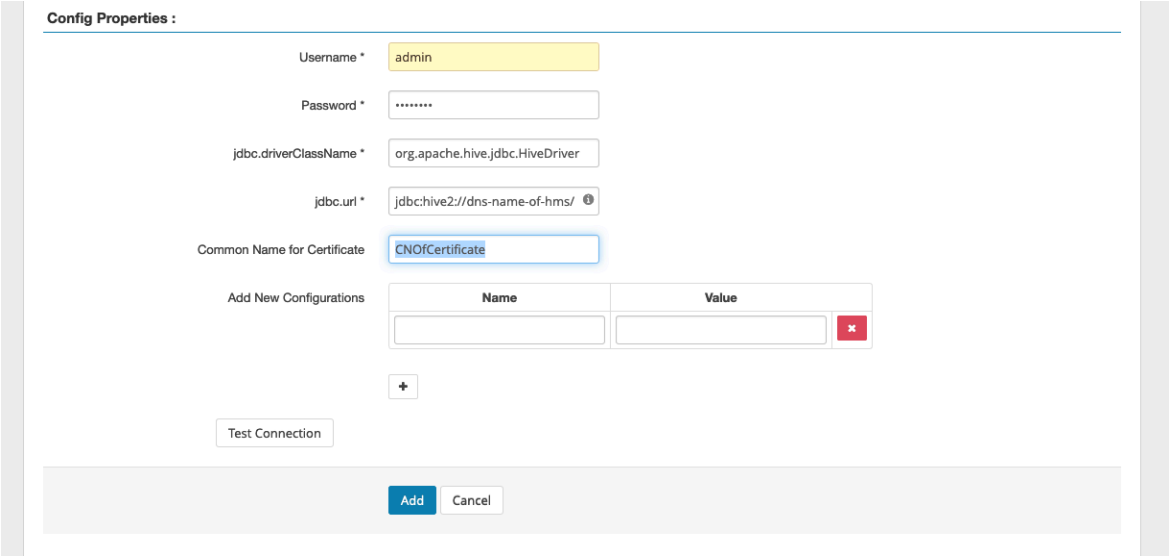
1. **Service Name (If displayed)**: Enter the service name. The suggested value is **`amazonemrhive`**. Make a note of this service name -- it's needed when creating an EMR security configuration.
2. **Display Name**: Enter the name to be displayed for the service. The suggested value is **`amazonemrhive`**.



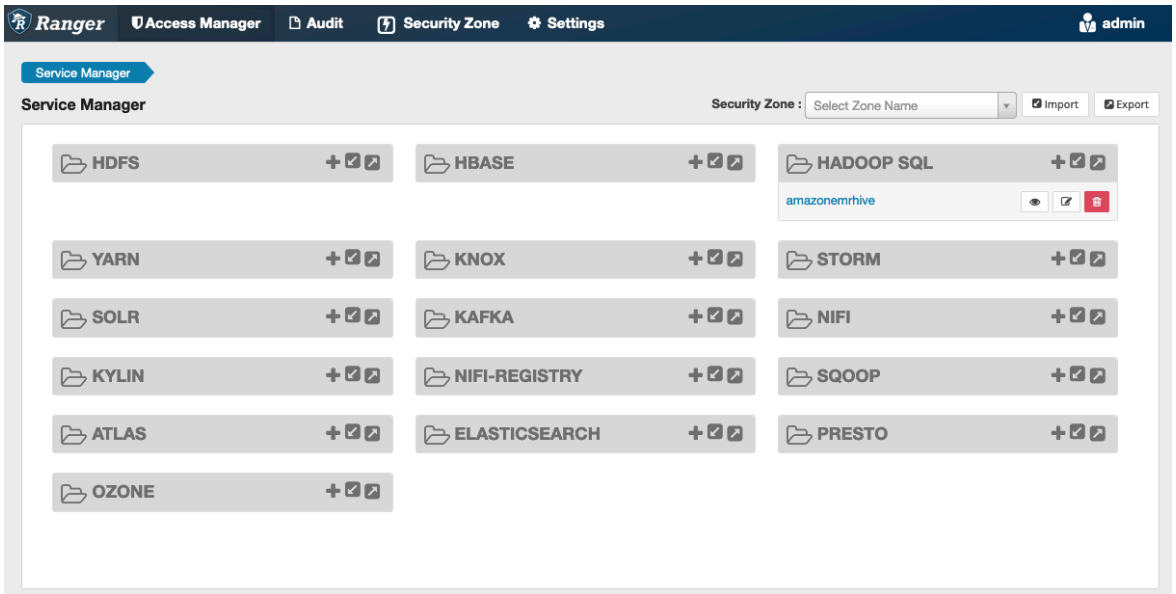
The Apache Hive Config Properties are used to establish a connection to your Apache Ranger Admin server with a HiveServer2 to implement auto complete when creating policies. The properties below are not required to be accurate if you do not have a persistent HiveServer2 process and can be filled with any information.

- **Username**: Enter a user name for the JDBC connection to an instance of an HiveServer2 instance.
- **Password**: Enter the password for the user name above.
- **jdbc.driver.ClassName**: Enter the class name of JDBC class for Apache Hive connectivity. The default value can be used.
- **jdbc.url**: Enter the JDBC connection string to use when connecting to HiveServer2.

- **Common Name for Certificate**: The CN field within the certificate used to connect to the admin server from a client plugin. This value must match the CN field in your TLS certificate that was created for the plugin.



The **Test Connection** button tests whether the values above can be used to successfully connect to the HiveServer2 instance. Once the service is successfully created, the Service Manager should look like below:



Considerations

Hive metadata server

The Hive metadata server can only be accessed by trusted engines, specifically Hive and `emr_record_server`, to protect against unauthorized access. The Hive metadata server is also accessed by all nodes on the cluster. The required port 9083 provides all nodes access to the main node.

Authentication

By default, Apache Hive is configured to authenticate using Kerberos as configured in the EMR Security configuration. HiveServer2 can be configured to authenticate users using LDAP as well. See Implementing LDAP authentication for Hive on a multi-tenant Amazon EMR cluster for information.

Limitations

The following are current limitations for the Apache Hive plugin on Amazon EMR 5.x:

- Hive roles are not currently supported. Grant, Revoke statements are not supported.
- Hive CLI is not supported. JDBC/Beeline is the only authorized way to connect Hive.
- `hive.server2.builtin.udf.blacklist` configuration should be populated with UDFs that you deem unsafe.

Apache Spark plugin

Amazon EMR has integrated EMR RecordServer to provide fine-grained access control for SparkSQL. EMR's RecordServer is a privileged process running on all nodes on an Apache Ranger-enabled cluster. When a Spark driver or executor runs a SparkSQL statement, all metadata and data requests go through the RecordServer. To learn more about EMR RecordServer, see the Amazon EMR components (p. 497) page.

Topics

- Supported features (p. 514)
- Redeploy service definition to use INSERT, ALTER, or DDL statements (p. 515)
- Installation of service definition (p. 515)
- Creating SparkSQL policies (p. 517)
- Considerations (p. 518)
- Limitations (p. 518)

Supported features

SQL statement/Ranger action	STATUS	Supported EMR release
SELECT	Supported	As of 5.32
SHOW DATABASES	Supported	As of 5.32
SHOW COLUMNS	Supported	As of 5.32
SHOW TABLES	Supported	As of 5.32
SHOW TABLE PROPERTIES	Supported	As of 5.32
DESCRIBE TABLE	Supported	As of 5.32
INSERT OVERWRITE	Supported	As of 5.34 and 6.4
INSERT INTO	Supported	As of 5.34 and 6.4
ALTER TABLE	Supported	As of 6.4
CREATE TABLE	Supported	As of 5.35 and 6.7
CREATE DATABASE	Supported	As of 5.35 and 6.7

SQL statement/Ranger action	STATUS	Supported EMR release
DROP TABLE	Supported	As of 5.35 and 6.7
DROP DATABASE	Supported	As of 5.35 and 6.7
DROP VIEW	Supported	As of 5.35 and 6.7
CREATE VIEW	Not Supported	

The following features are supported when using SparkSQL:

- Fine-grained access control on tables within the Hive Metastore, and policies can be created at a database, table, and column level.
- Apache Ranger policies can include grant policies and deny policies to users and groups.
- Audit events are submitted to CloudWatch Logs.

Redeploy service definition to use INSERT, ALTER, or DDL statements

Note

Starting with Amazon EMR 6.4, you can use Spark SQL with the statements: INSERT INTO, INSERT OVERWRITE, or ALTER TABLE. Starting with Amazon EMR 6.7, you can use Spark SQL to create or drop databases and tables. If you have an existing installation on Apache Ranger server with Apache Spark service definitions deployed, use the following code to redeploy the service definitions.

```
# Get existing Spark service definition id calling Ranger REST API and JSON
 processor
curl --silent -f -u <admin_user_login>:<password_for_ranger_admin_user> \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
-k 'https://*<RANGER SERVER ADDRESS>*:6182/service/public/v2/api/servicedef/name/
amazon-emr-spark' | jq .id

# Download the latest Service definition
wget https://s3.amazonaws.com/elasticmapreduce/ranger/service-definitions/
version-2.0/ranger-servicedef-amazon-emr-spark.json

# Update the service definition using the Ranger REST API
curl -u <admin_user_login>:<password_for_ranger_admin_user> -X PUT -d @ranger-
servicedef-amazon-emr-spark.json \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
-k 'https://*<RANGER SERVER ADDRESS>*:6182/service/public/v2/api/servicedef/<Spark
 service definition id from step 1>'
```

Installation of service definition

The installation of EMR's Apache Spark service definition requires the Ranger Admin server to be setup. See Set up Ranger Admin server (p. 499).

Follow these steps to install the Apache Spark service definition:

Step 1: SSH into the Apache Ranger Admin server

For example:

```
ssh ec2-user@ip-xxx-xxx-xxx-xxx.ec2.internal
```

Step 2: Download the service definition and Apache Ranger Admin server plugin

In a temporary directory, download the service definition. This service definition is supported by Ranger 2.x versions.

```
mkdir /tmp/emr-spark-plugin/
cd /tmp/emr-spark-plugin/

wget https://s3.amazonaws.com/elasticmapreduce/ranger/service-definitions/version-2.0/
ranger-spark-plugin-2.x.jar
wget https://s3.amazonaws.com/elasticmapreduce/ranger/service-definitions/version-2.0/
ranger-servicedef-amazon-emr-spark.json
```

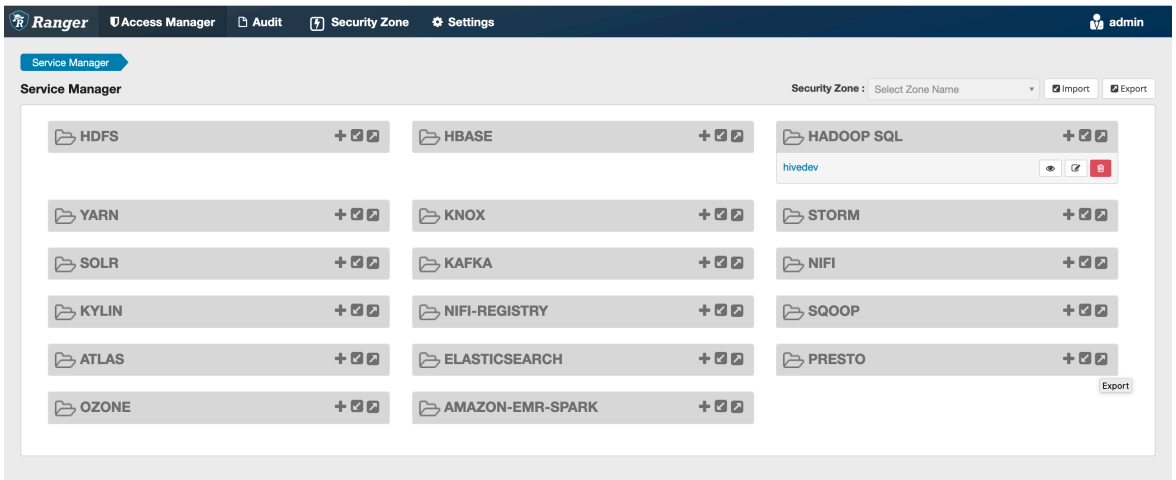
Step 3: Install the Apache Spark plugin for Amazon EMR

```
export RANGER_HOME=.. # Replace this Ranger Admin's home directory eg /usr/lib/ranger/
ranger-2.0.0-admin
mkdir $RANGER_HOME/ews/webapp/WEB-INF/classes/ranger-plugins/amazon-emr-spark
mv ranger-spark-plugin-2.x.jar $RANGER_HOME/ews/webapp/WEB-INF/classes/ranger-plugins/
amazon-emr-spark
```

Step 4: Register the Apache Spark service definition for Amazon EMR

```
curl -u *<admin users login>*:*_<_**_password_ **_for_** _ranger admin user_**_>_* -X POST
 -d @ranger-servicedef-amazon-emr-spark.json \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
-k 'https://*<RANGER SERVER ADDRESS>*:6182/service/public/v2/api/servicedef'
```

If this command runs successfully, you see a new service in your Ranger Admin UI called "AMAZON-EMR-SPARK", as shown in the following image (Ranger version 2.0 is shown).

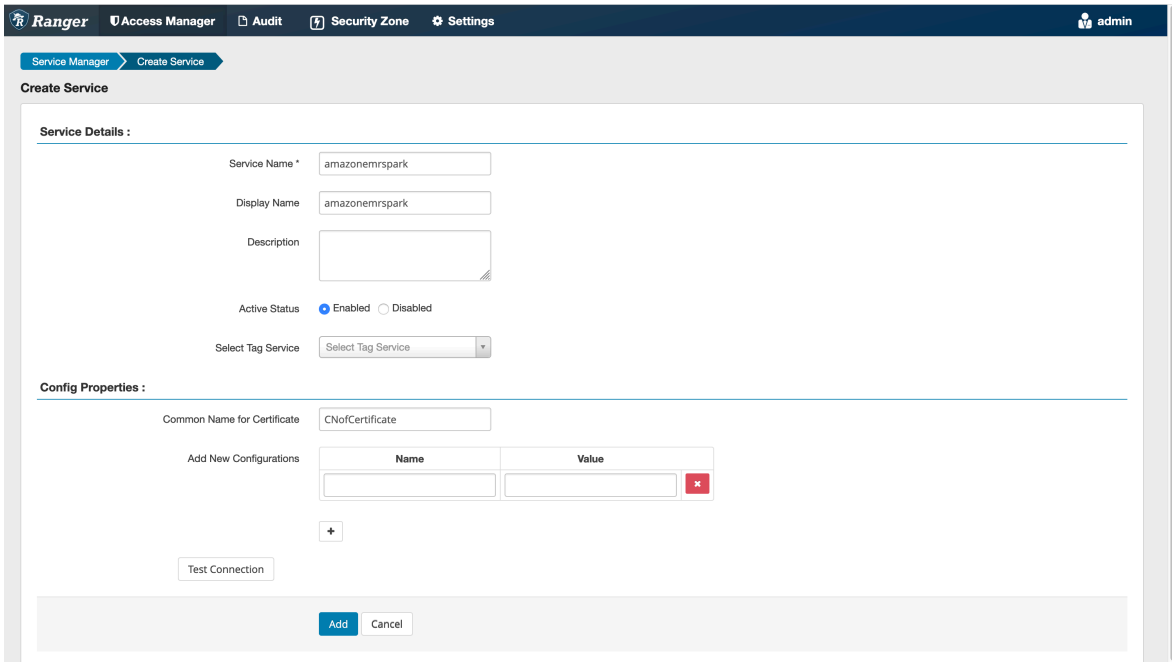


Step 5: Create an instance of the AMAZON-EMR-SPARK application

Service Name (If displayed): The service name that will be used. The suggested value is **`amazonemrspark`**. Note this service name as it will be needed when creating an EMR security configuration.

Display Name: The name to be displayed for this instance. The suggested value is **`amazonemrspark`**.

Common Name For Certificate: The CN field within the certificate used to connect to the admin server from a client plugin. This value must match the CN field in your TLS certificate that was created for the plugin.



Note
The TLS certificate for this plugin should have been registered in the trust store on the Ranger Admin server. See TLS certificates (p. 500) for more details.

Creating SparkSQL policies

When creating a new policy, the fields to fill in are:

Policy Name: The name of this policy.

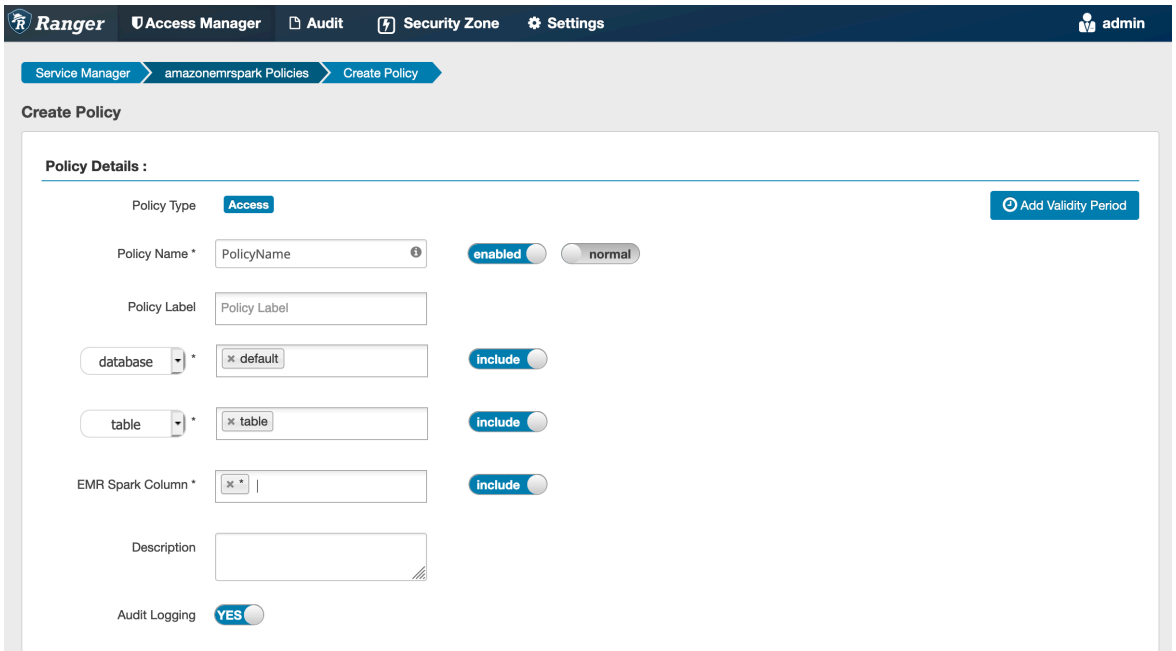
Policy Label: A label that you can put on this policy.

Database: The database that this policy applies to. The wildcard "*" represents all databases.

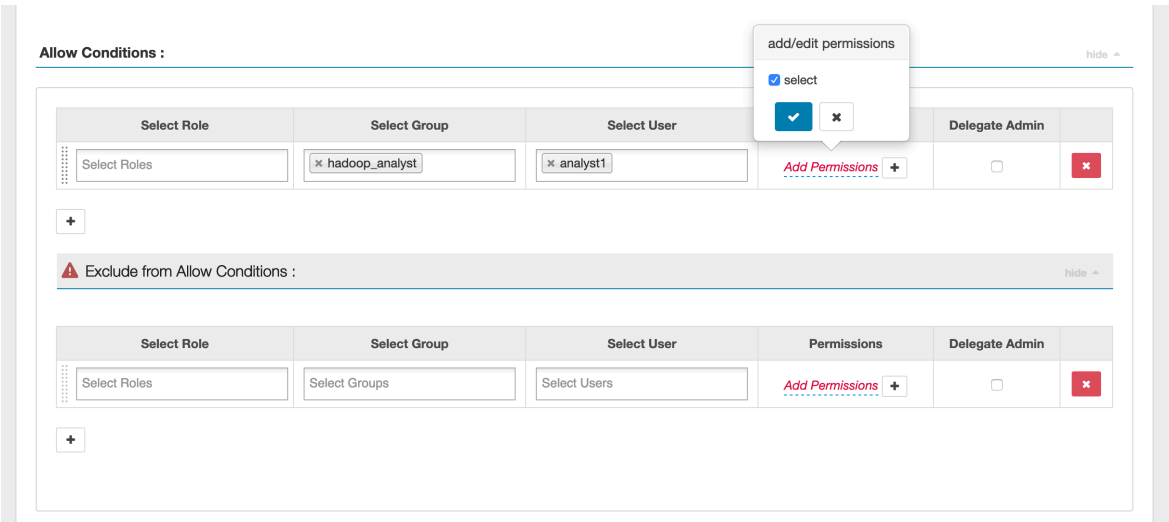
Table: The tables that this policy applies to. The wildcard "*" represents all tables.

EMR Spark Column: The columns that this policy applies to. The wildcard "*" represents all columns.

Description: A description of this policy.



To specify the users and groups, enter the users and groups below to grant permissions. You can also specify exclusions for the **allow** conditions and **deny** conditions.



After specifying the allow and deny conditions, click **Save**.

Considerations

Each node within the EMR cluster must be able to connect to the main node on port 9083.

Limitations

The following are current limitations for the Apache Spark plugin:

- Record Server will always connect to HMS running on an Amazon EMR cluster. Configure HMS to connect to Remote Mode, if required. You should not put config values inside the Apache Spark Hive-site.xml configuration file.

- Tables created using Spark datasources on CSV or Avro are not readable using EMR RecordServer. Use Hive to create and write data, and read using Record.
- Delta Lake and Hudi tables are not supported.
- Users must have access to the default database. This is a requirement for Apache Spark.
- Ranger Admin server does not support auto-complete.
- The SparkSQL plugin for Amazon EMR does not support row filters or data masking.
- When using ALTER TABLE with Spark SQL, a partition location must be the child directory of a table location. Inserting data into a partition where the partition location is different from the table location is not supported.

EMRFS S3 plugin

To make it easier to provide access controls against objects in S3 on a multi-tenant cluster, the EMRFS S3 plugin provides access controls to the data within S3 when accessing it through EMRFS. You can allow access to S3 resources at a user and group level.

To achieve this, when your application attempts to access data within S3, EMRFS sends a request for credentials to the Secret Agent process, where the request is authenticated and authorized against an Apache Ranger plugin. If the request is authorized, then the Secret Agent assumes the IAM role for Apache Ranger Engines with a restricted policy to generate credentials that only have access to the Ranger policy that allowed the access. The credentials are then passed back to EMRFS to access S3.

Topics

- Supported features (p. 519)
- Installation of service configuration (p. 519)
- Creating EMRFS S3 policies (p. 521)
- EMRFS S3 policies usage notes (p. 522)
- Limitations (p. 525)

Supported features

EMRFS S3 plugin provides storage level authorization. Policies can be created to provide access to users and groups to S3 buckets and prefixes. Authorization is done only against EMRFS.

Installation of service configuration

The installation of the Trino service definition requires that the Ranger Admin server be set up. To set up the Ranger Admin sever, see Set up Ranger Admin server (p. 499).

Follow these steps to install the EMRFS service definition.

Step 1: SSH into the Apache Ranger Admin server.

For example:

```
ssh ec2-user@ip-xxx-xxx-xxx-xxx.ec2.internal
```

Step 2: Download the Amazon EMR service definition and Apache Ranger Admin server plugin.

In a temporary directory, download the Amazon EMR service definition. This service definition is supported by Ranger 2.x versions.

```
mkdir /tmp/emr-emrfs-plugin/
cd /tmp/emr-emrfs-plugin/
```

```
wget https://s3.amazonaws.com/elasticmapreduce/ranger/service-definitions/version-2.0/
ranger-servicedef-amazon-emr-emrfs.json
wget https://s3.amazonaws.com/elasticmapreduce/ranger/service-definitions/version-2.0/
ranger-emr-emrfs-plugin-2.x.jar
```

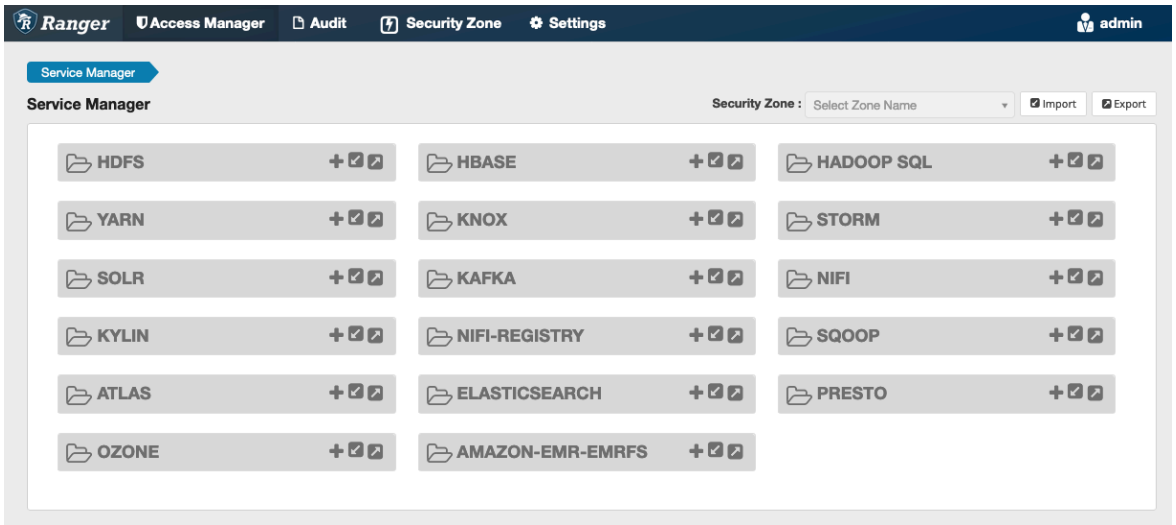
Step 3: Install the Apache EMRFS S3 plugin for Amazon EMR.

```
export RANGER_HOME=.. # Replace this Ranger Admin's home directory eg /usr/lib/ranger/
ranger-2.0.0-admin
mkdir $RANGER_HOME/ews/webapp/WEB-INF/classes/ranger-plugins/amazon-emr-emrfs
mv ranger-emr-emrfs-plugin-2.x.jar $RANGER_HOME/ews/webapp/WEB-INF/classes/ranger-plugins/
amazon-emr-emrfs
```

Step 4: Register EMRFS S3 service definition.

```
curl -u *<admin users login>*:*_<_**_password_ **_for_** _ranger admin user_**_>_* -X POST
 -d @ranger-servicedef-amazon-emr-emrfs.json \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
-k 'https://*<RANGER SERVER ADDRESS>*:6182/service/public/v2/api/servicedef'
```

If this command runs successfully, you see a new service in the Ranger Admin UI called "AMAZON-EMR-S3", as shown in the following image (Ranger version 2.0 is shown).



Step 5: Create an instance of the AMAZON-EMR-EMRFS application.

Create an instance of the service definition.

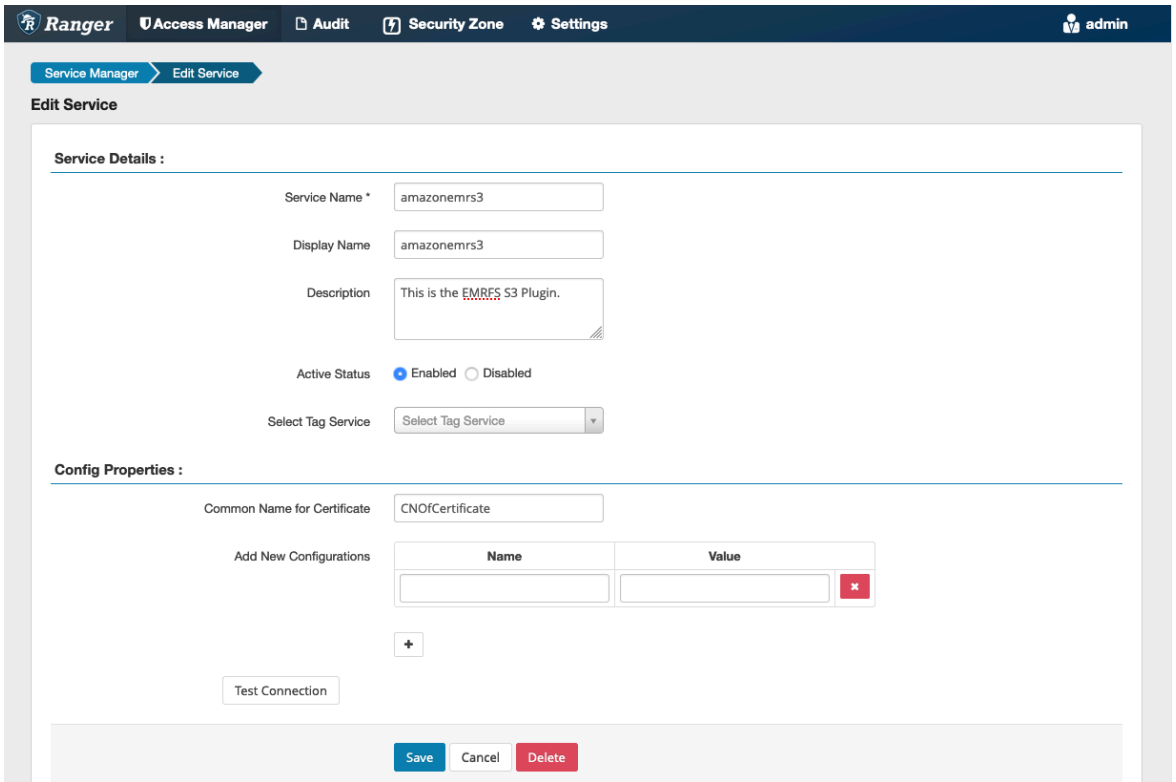
- Click on the **+** next to AMAZON-EMR-EMRFS.

Fill in the following fields:

Service Name (If displayed): The suggested value is **`amazonemrspark`**. Note this service name as it will be needed when creating an EMR security configuration.

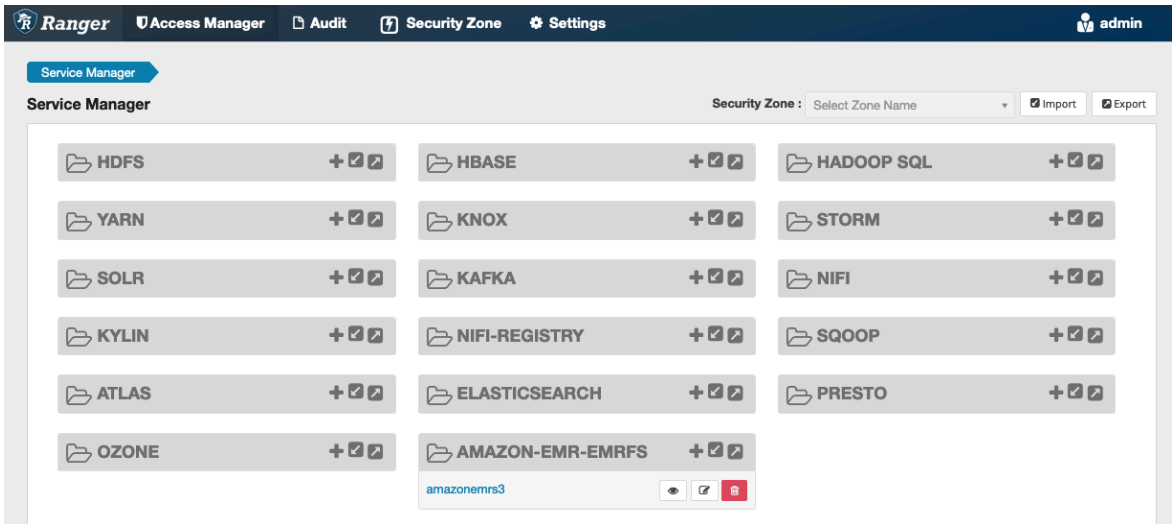
Display Name: The name displayed for this service. The suggested value is **`amazonemrspark`**.

Common Name For Certificate: The CN field within the certificate used to connect to the admin server from a client plugin. This value must match the CN field in the TLS certificate that was created for the plugin.



> **Note**
> The TLS certificate for this plugin should have been registered in the trust store on the Ranger Admin server. See TLS certificates (p. 500) for more details.

When the service is created, the Service Manager includes "AMAZON-EMR-EMRFS", as shown in the following image.



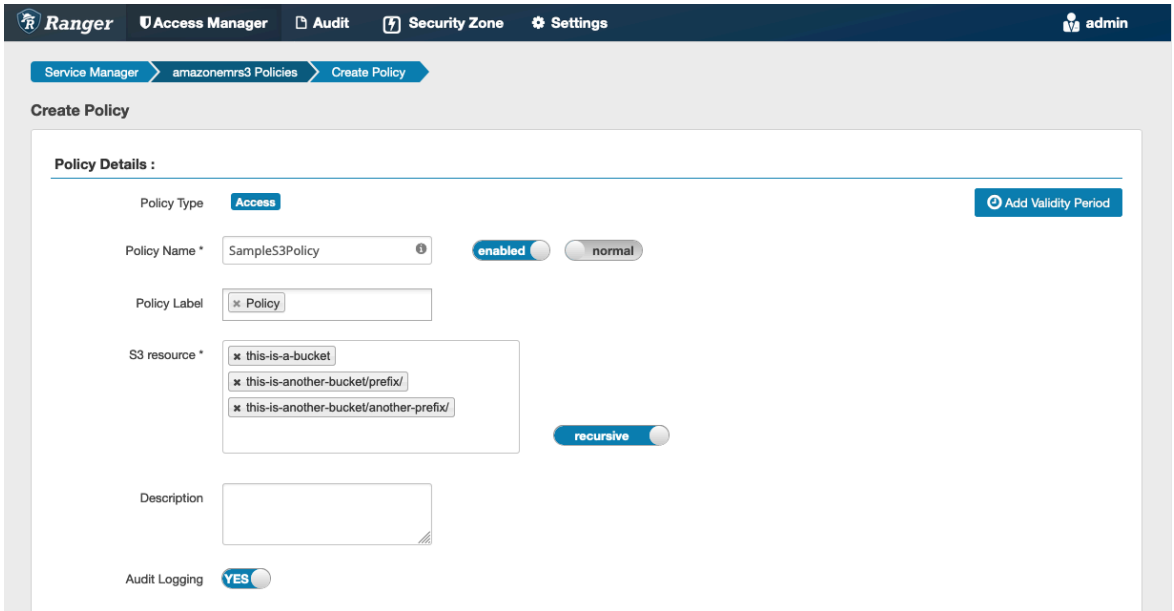
Creating EMRFS S3 policies

To create a new policy in the **Create policy** page of the Service Manager, fill in the following fields.

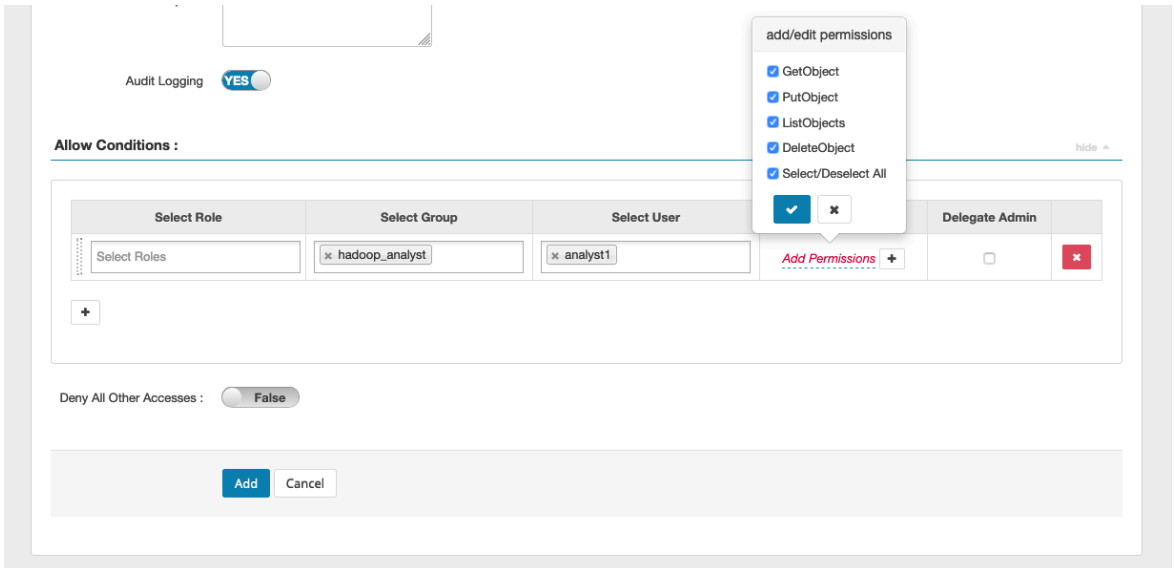
Policy Name: The name of this policy.

Policy Label: A label that you can put on this policy.

S3 Resource: A resource starting with the bucket and optional prefix. See EMRFS S3 policies usage notes (p. 522) for information on best practices. Resources in Ranger Admin server should not contain `s3://`, `s3a://` or `s3n://`.



You can specify users and groups to grant permissions. You can also specify exclusions for **allow** conditions and **deny** conditions.



> **Note**
> A maximum of three resources are allowed for each policy. Adding more than three resources may result in an error when this policy is used on an EMR cluster. Adding more than three policies displays a reminder about the policy limit.

EMRFS S3 policies usage notes

When creating S3 policies within Apache Ranger, there are some usage considerations to be aware of.

Permissions to multiple S3 objects

You can use recursive policies and wildcard expressions to give permissions to multiple S3 objects with common prefixes. Recursive policies give permissions to all objects with a common prefix. Wildcard expressions select multiple prefixes. Together, they give permissions to all objects with multiple common prefixes as shown in the following examples.

Example Using a recursive policy

Suppose you want permissions to list all the parquet files in an S3 bucket organized as follows.

```
s3://sales-reports/americas/
    +- year=2000
    |      +- data-q1.parquet
    |      +- data-q2.parquet
    +- year=2019
    |      +- data-q1.json
    |      +- data-q2.json
    |      +- data-q3.json
    |      +- data-q4.json
    |
    +- year=2020
    |      +- data-q1.parquet
    |      +- data-q2.parquet
    |      +- data-q3.parquet
    |      +- data-q4.parquet
    |      +- annual-summary.parquet
    +- year=2021
```

First, consider the parquet files with the prefix `s3://sales-reports/americas/year=2000`. You can grant GetObject permissions to all of them in two ways:

Using non-recursive policies: One option is to use two separate non-recursive policies, one for the directory and the other for the files.

The first policy grants permission to the prefix `s3://sales-reports/americas/year=2020` (there is no trailing /).

```
- S3 resource = "sales-reports/americas/year=2000"
- permission = "GetObject"
- user = "analyst"
```

The second policy uses wildcard expression to grant permissions all the files with prefix `sales-reports/americas/year=2020/` (note the trailing /).

```
- S3 resource = "sales-reports/americas/year=2020/*"
- permission = "GetObject"
- user = "analyst"
```

Using a recursive policy: A more convenient alternative is to use a single recursive policy and grant recursive permission to the prefix.

```
 - S3 resource = "sales-reports/americas/year=2020"
 - permission = "GetObject"
 - user = "analyst"
 - is recursive = "True"
```

So far, only the parquet files with the prefix `s3://sales-reports/americas/year=2000` have been included. You can now also include the parquet files with a different prefix, `s3://sales-reports/americas/year=2020`, into the same recursive policy by introducing a wildcard expression as follows.

```
- S3 resource = "sales-reports/americas/year=20?0"
- permission = "GetObject"
- user = "analyst"
- is recursive = "True"
```

Policies for PutObject and DeleteObject permissions

Writing policies for `PutObject` and `DeleteObject` permissions to files on EMRFS need special care because, unlike GetObject permissions, they need additional recursive permissions granted to the prefix.

Example Policies for PutObject and DeleteObject permissions

For example, deleting the file `annual-summary.parquet` requires not only a DeleteObject permission to the actual file.

```
- S3 resource = "sales-reports/americas/year=2020/annual-summary.parquet"
- permission = "DeleteObject"
- user = "analyst"
```

It also requires a policy granting recursive `GetObject` and `PutObject` permissions to its prefix.

Similarly, modifying the file `annual-summary.parquet`, requires not only a `PutObject` permission to the actual file.

```
- S3 resource = "sales-reports/americas/year=2020/annual-summary.parquet"
- permission = "PutObject"
- user = "analyst"
```

It also requires a policy granting recursive `GetObject` permission to its prefix.

```
- S3 resource = "sales-reports/americas/year=2020"
- permission = "GetObject"
- user = "analyst"
- is recursive = "True"
```

Wildcards in policies

There are two areas in which wildcards can be specified. When specifying an S3 resource, the "*" and "?" can be used. The "*" provides matching against an S3 path and matches everything after the prefix. For example, the following policy.

```
S3 resource = "sales-reports/americas/*"
```

This matches the following S3 paths.

```
sales-reports/americas/year=2020/
sales-reports/americas/year=2019/
sales-reports/americas/year=2019/month=12/day=1/afile.parquet
sales-reports/americas/year=2018/month=6/day=1/afile.parquet
sales-reports/americas/year=2017/afile.parquet
```

The "?" wildcard matches only a single character. For example, for the policy.

```
S3 resource = "sales-reports/americas/year=201?/"
```

This matches the following S3 paths.

```
sales-reports/americas/year=2019/
sales-reports/americas/year=2018/
sales-reports/americas/year=2017/
```

Wildcards in users

There are two built-in wildcards when assigning users to provide access to users. The first is the "{USER}" wildcard that provides access to all users. The second wildcard is "{OWNER}", which provides access to the owner of a particular object or directly. However, the "{USER}" wildcard is currently not supported.

Limitations

The following are current limitations of the EMRFS S3 plugin:

- Apache Ranger policies can have at most three policies.
- Access to S3 must be done through EMRFS and can be used with Hadoop-related applications. The following is not supported:

  - Boto3 libraries

  - AWS SDK and AWK CLI

  - S3A open source connector
- Apache Ranger deny policies are not supported.
- Operations on S3 with keys having CSE-KMS encryption are currently not supported.
- Cross-Region support is not supported.
- Apache Ranger's Security Zone feature is not supported. Access control restrictions defined using the Security Zone feature are not applied on your Amazon EMR clusters.
- The Hadoop user does not generate any audit events as Hadoop always accesses the EC2 Instance Profile.
- It's recommended that you disable Amazon EMR Consistency View. S3 is strongly consistent, so it's no longer needed. See Amazon S3 strong consistency for more information.
- The EMRFS S3 plugin makes numerous STS calls. It's advised that you do load testing on a development account and monitor STS call volume. It is also recommended that you make an STS request to raise AssumeRole service limits.

Trino plugin

Trino (previously PrestoSQL) is a SQL query engine that you can use to run queries on data sources such as HDFS, object storage, relational databases, and NoSQL databases. It eliminates the need to migrate data into a central location and allows you to query the data from whenever it sits. Amazon EMR provides an Apache Ranger plugin to provide fine-grained access controls for Trino. The plugin is compatible with open source Apache Ranger Admin server version 2.0 and later.

Topics

- Supported features (p. 526)
- Installation of service configuration (p. 526)
- Creating Trino policies (p. 528)
- Considerations (p. 529)
- Limitations (p. 530)

Supported features

The Apache Ranger plugin for Trino on Amazon EMR supports all the functionality of the Trino query engine that is protected by fine-grained access control. This includes database, table, column level access controls and row filtering and data masking. Apache Ranger policies can include grant policies and deny policies to users and groups. Audit events are also submitted to CloudWatch logs.

Installation of service configuration

The installation of the Trino service definition requires that the Ranger Admin server be set up. To set up the Ranger Admin sever, see Set up Ranger Admin server (p. 499).

Follow these steps to install the Trino service definition.

1. SSH into the Apache Ranger Admin server.

```
ssh ec2-user@ip-xxx-xxx-xxx-xxx.ec2.internal
```

2. Uninstall the Presto server plugin, if it exists. Run the following command. If this errors out with a “Service not found” error, this means the Presto server plugin wasn't installed on your server. Proceed to the next step.

```
curl -f -u *<admin users login>*:*_<_**_password_ **_for_** _ranger admin user_**_>_*
 -X DELETE -k 'https://*<RANGER SERVER ADDRESS>*:6182/service/public/v2/api/servicedef/
name/presto'
```

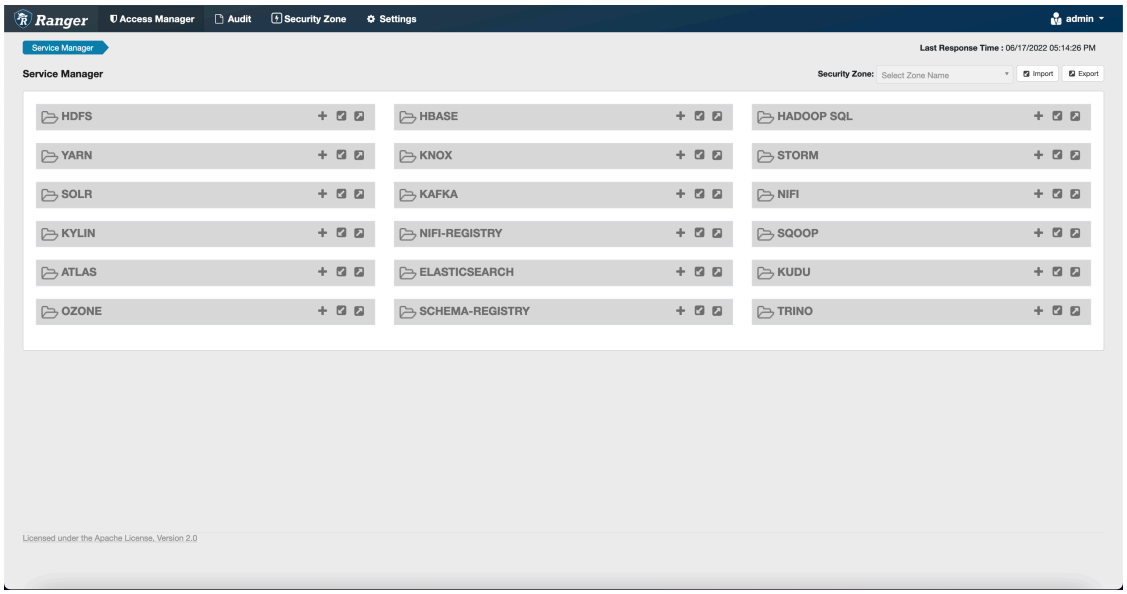
3. Download the service definition and Apache Ranger Admin server plugin. In a temporary directory, download the service definition. This service definition is supported by Ranger 2.x versions.

```
wget https://s3.amazonaws.com/elasticmapreduce/ranger/service-definitions/version-2.0/
ranger-servicedef-amazon-emr-trino.json
```

4. Register the Apache Trino service definition for Amazon EMR.

```
curl -u *<admin users login>*:*_<_**_password_ **_for_** _ranger admin user_**_>_* -X
 POST -d @ranger-servicedef-amazon-emr-trino.json \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
-k 'https://*<RANGER SERVER ADDRESS>*:6182/service/public/v2/api/servicedef'
```

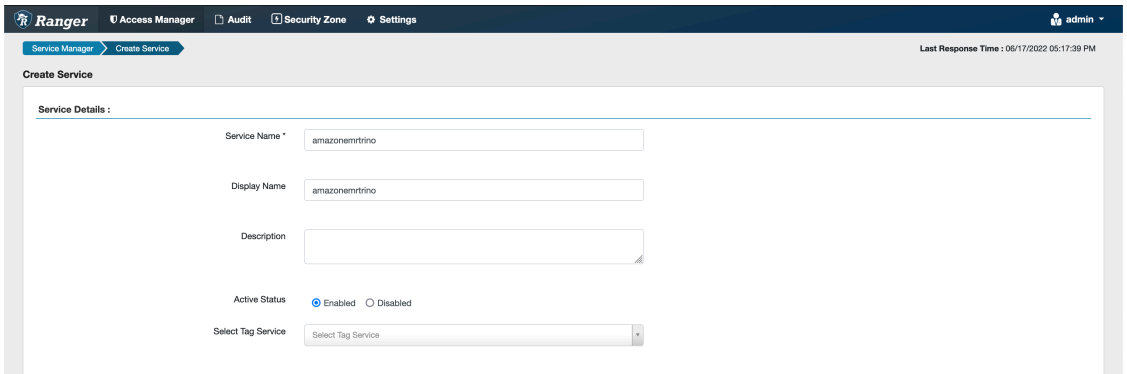
If this command runs successfully, you see a new service in your Ranger Admin UI called `TRINO`, as shown in the following image.



5. Create an instance of the `TRINO` application, entering the following information.

   Service Name: The service name that you'll use. The suggested value is `amazonemrtrino`. Note this service name, as it will be needed when creating an Amazon EMR security configuration.

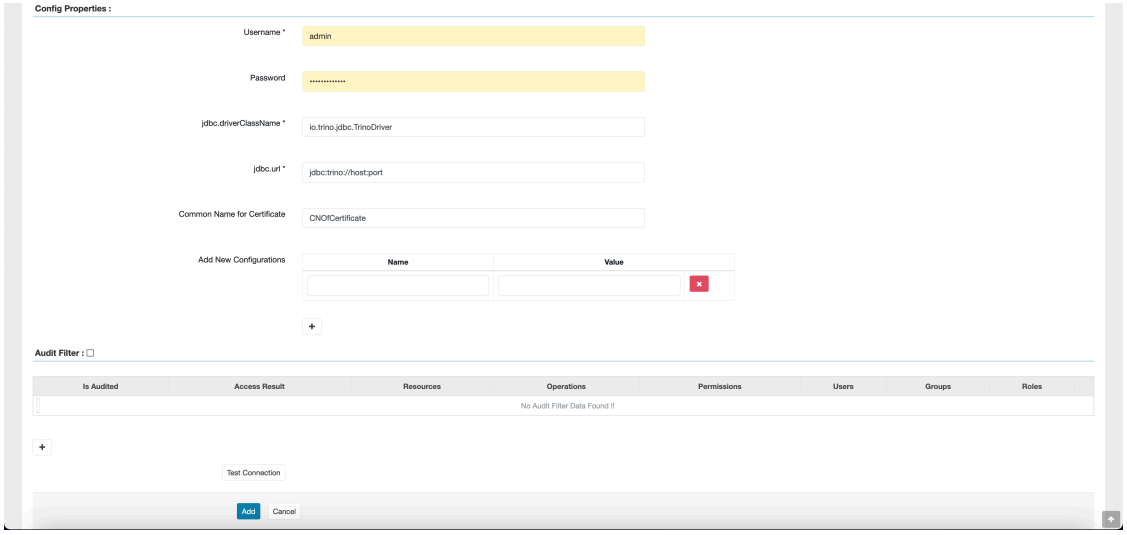
   Display Name: The name to be displayed for this instance. The suggested value is `amazonemrtrino`.



   jdbc.driver.ClassName: The class name of JDBC class for Trino connectivity. You can use the default value.

   jdbc.url: The JDBC connection string to use when connecting to Trino coordinator.

   Common Name For Certificate: The CN field within the certificate used to connect to the admin server from a client plugin. This value must match the CN field in your TLS certificate that was created for the plugin.



Note that the TLS certificate for this plugin should have been registered in the trust store on the Ranger Admin server. For more information, see TLS certificates.

Creating Trino policies

When you create a new policy, fill in the following fields.

Policy Name: The name of this policy.

Policy Label: A label that you can put on this policy.

Catalog: The catalog that this policy applies to. The wildcard "*" represents all catalogs.

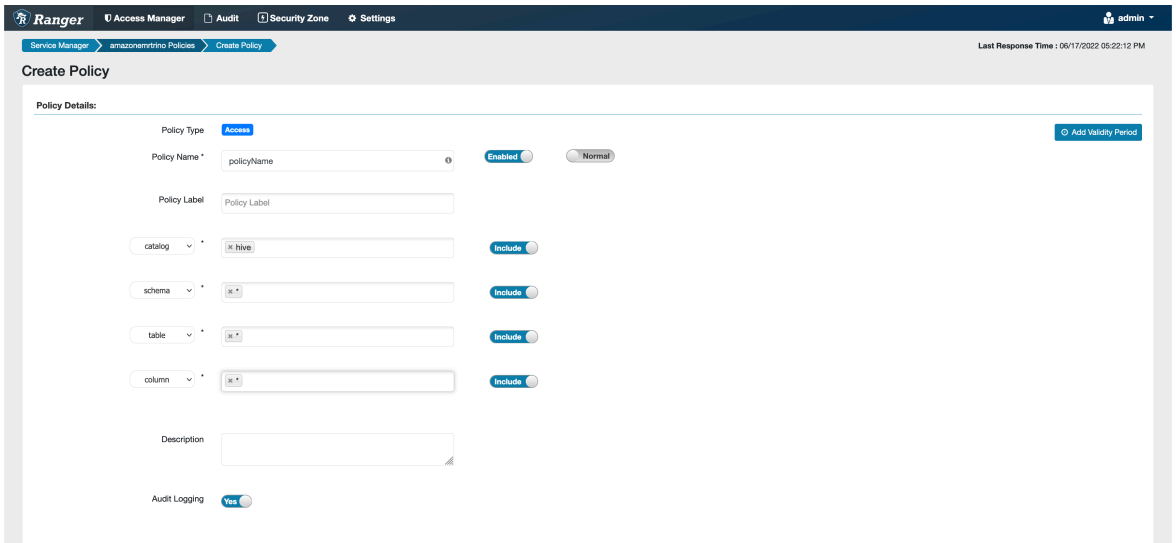
Schema: The schemas that this policy applies to. The wildcard "*" represents all schemas.

Table: The tables that this policy applies to. The wildcard "*" represents all tables.

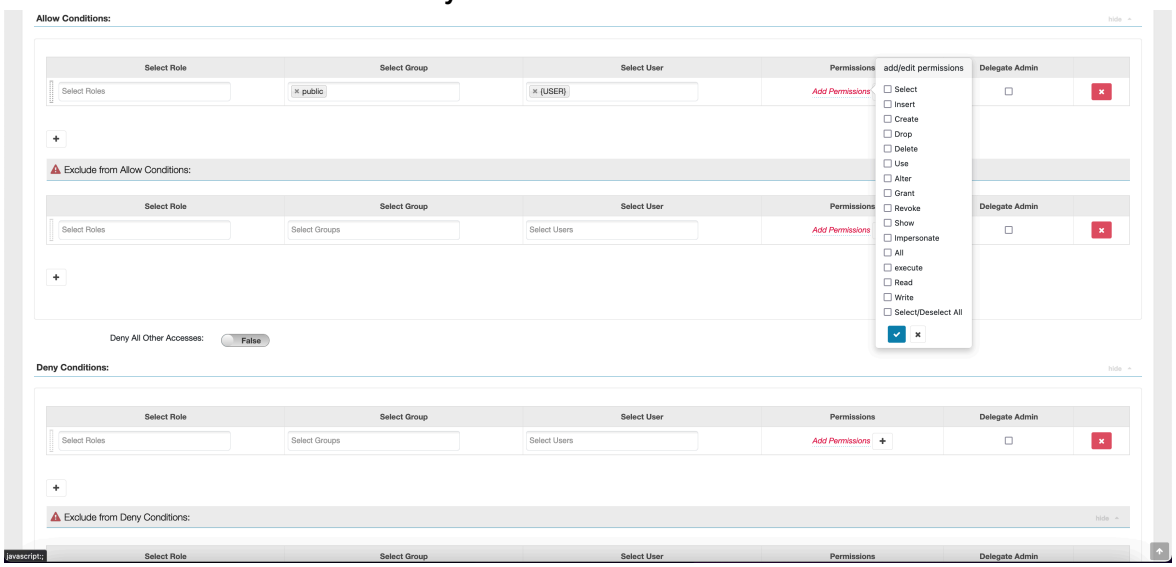
Column: The columns that this policy applies to. The wildcard "*" represents all columns.

Description: A description of this policy.

Other types of policies exist for the **Trino User** (for user impersonation access), the **Trino System/Session Property** (for altering engine system or session properties), **Functions/Procedures** (for allowing function or procedure calls), and the **URL** (for granting read/write access to the engine on data locations).



To grant permissions to specific users and groups, enter the users and groups. You can also specify exclusions for **allow** conditions and **deny** conditions.



After specifying the allow and deny conditions, choose **Save**.

Considerations

When creating Trino policies within Apache Ranger, there are some usage considerations to be aware of.

Hive metadata server

The Hive metadata server can only be accessed by trusted engines, specifically the Trino engine, to protect against unauthorized access. The Hive metadata server is also accessed by all nodes on the cluster. The required port 9083 provides all nodes access to the main node.

Authentication

By default, Trino is configured to authenticate using Kerberos as configured in the Amazon EMR security configuration.

In-transit encryption required

The Trino plugin requires you to have in-transit encryption enabled in the Amazon EMR security configuration. To enable encryption, see Encryption in transit (p. 377).

Limitations

The following are current limitations of the Trino plugin:

- Ranger Admin server doesn't support auto-complete.

Apache Ranger troubleshooting

Here are some commonly diagnosed issues related to using Apache Ranger.

Recommendations

- **Test using a single main node cluster:** Single node master clusters provision quicker than a multi-node cluster which can decrease the time for each testing iteration.
- **Set development mode on the cluster.** When starting your EMR cluster, set the `--additional-info"` parameter to:

  ```
  '{"clusterType":"development"}'
  ```

  This parameter can only be set through the AWS CLI or AWS SDK and is not available through the Amazon EMR console. When this flag is set, and the master fails to provision, the Amazon EMR service keeps the cluster alive for some time before it decommissions it. This time is very useful for probing various log files before the cluster is terminated.

EMR cluster failed to provision

There are several reasons why an Amazon EMR cluster may fail to start. Here are a few ways to diagnose the issue.

Check EMR provisioning logs

Amazon EMR uses Puppet to install and configure applications on a cluster. Looking at the logs will provide details as to if there are any errors during the provisioning phase of a cluster. The logs are accessible on cluster or S3 if logs are configured to be pushed to S3.

The logs are stored in `/var/log/provision-node/apps-phase/0/{UUID}/puppet.log` on the disk and `s3://<LOG LOCATION>/<CLUSTER ID>/node/<EC2 INSTANCE ID>/provision-node/apps-phase/0/{UUID}/puppet.log.gz`.

Common Error Messages

Error message	Cause
Puppet (err): Systemd start for emr-record-server failed! journalctl log for emr-record-server:	EMR Record Server failed to start. See EMR Record Server logs below.
Puppet (err): Systemd start for emr-record-server failed! journalctl log for emrsecretagent:	EMR Secret Agent failed to start. See Check Secret Agent logs below.
/Stage[main]/ Ranger_plugins::Ranger_hive_plugin/	The private TLS certificate in Secret Manager for the Apache Ranger plugin certificate is not in the

Error message	Cause
Ranger_plugins::Prepare_two_way_tls[configure 2-way TLS in Hive plugin]/Exec[create keystore and truststore for Ranger Hive plugin]/returns (notice): 140408606197664:error:0906D06C:PEM routines:PEM_read_bio:no start line:pem_lib.c:707:Expecting: ANY PRIVATE KEY	correct format or is not a private certificate. See TLS certificates (p. 500) for certificate formats.
/Stage[main]/Ranger_plugins::Ranger_s3_plugin/Ranger_plugins::Prepare_two_way_tls[configure 2-way TLS in Ranger s3 plugin]/Exec[create keystore and truststore for Ranger amazon-emr-s3 plugin]/returns (notice): An error occurred (AccessDeniedException) when calling the GetSecretValue operation: User: arn:aws:sts::XXXXXXXXXXX:assumed-role/EMR_EC2_DefaultRole/i-XXXXXXXXXXXX is not authorized to perform: secretsmanager:GetSecretValue on resource: arn:aws:secretsmanager:us-east-1:XXXXXXXXXX:secret:AdminServer-XXXXX	The EC2 Instance profile role does not have the correct permissions to retrieve the TLS certificates from Secrets Agent.

Check SecretAgent logs

Secret Agent logs are located at `/emr/secretagent/log/` on an EMR node, or in the `s3://<LOG LOCATION>/<CLUSTER ID>/node/<EC2 INSTANCE ID>/daemons/secretagent/` directory in S3.

Common Error Messages

Error message	Cause
Exception in thread "main" com.amazonaws.services.securitytoken.model.AWSSecurityTokenServiceException: User: arn:aws:sts::XXXXXXXXXXXX:assumed-role/EMR_EC2_DefaultRole/i-XXXXXXXXXXXXXXX is not authorized to perform: sts:AssumeRole on resource: arn:aws:iam::XXXXXXXXXXXX:role/*RangerPluginDataAccessRole* (Service: AWSSecurityTokenService; Status Code: 403; Error Code: AccessDenied; Request ID: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX; Proxy: null)	The above exception means that the EMR EC2 instance profile role does not have permissions to assume the role **RangerPluginDataAccessRole**. See IAM roles for native integration with Apache Ranger (p. 502).
ERROR qtp54617902-149: Web App Exception Occurred javax.ws.rs.NotAllowedException: HTTP 405 Method Not Allowed	These errors can be safely ignored.

Check Record Server Logs (for SparkSQL)

EMR Record Server logs are available at /var/log/emr-record-server/ on an EMR node, or they can be found in the s3://<LOG LOCATION>/<CLUSTER ID>/node/<EC2 INSTANCE ID>/daemons/emr-record-server/ directory in S3.

Common Error Messages

Error message	Cause
InstanceMetadataServiceResourceFetcher:105 - [] Fail to retrieve token com.amazonaws.SdkClientException: Failed to connect to service endpoint	The EMR SecretAgent failed to come up or is having an issue. Inspect the SecretAgent logs for errors and the puppet script to determine if there were any provisioning errors.

Queries are unexpectedly failing

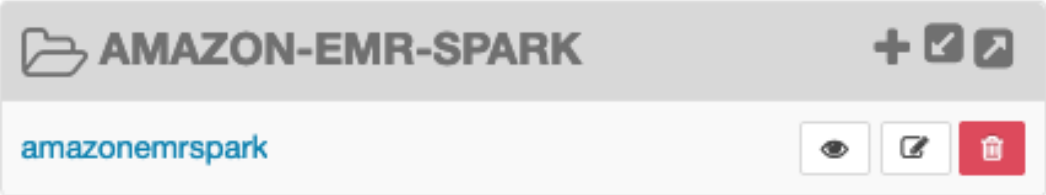
Check Apache Ranger plugin logs (Apache Hive, EMR RecordServer, EMR SecretAgent, etc., logs)

This section is common across all applications that integrate with the Ranger plugin, such as Apache Hive, EMR Record Server, and EMR SecretAgent.

Common Error Messages

Error message	Cause
ERROR PolicyRefresher:272 - [] PolicyRefresher(serviceName=policy-repository): failed to find service. Will clean up local cache of policies (-1)	This error messages means that the service name you provided in the EMR security configuration does not match a service policy repository in the Ranger Admin server.

If within Ranger Admin server your AMAZON-EMR-SPARK service looks like the following, then you should enter **`amazonemrspark`** as the service name.



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 in the *Amazon EC2 User Guide 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 that Amazon EMR creates, 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 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.

> **Important**
> 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.

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 to open ports. Be sure to add rules that only allow traffic from trusted and authenticated clients for the protocols and ports that are required to run your workloads.

You can configure Amazon EMR block public access in each region that you use to prevent cluster creation if a rule allows public access on any port that you don't add to a list of exceptions. For AWS accounts created after July 2019, Amazon EMR block public access is on by default. For AWS accounts that created a cluster before July 2019, Amazon EMR block public access is off by default. For more information, see Using Amazon EMR block public access (p. 542).

Topics

- Working with Amazon EMR-managed security groups (p. 533)
- Working with additional security groups (p. 539)
- Specifying Amazon EMR-managed and additional security groups (p. 539)
- Specifying EC2 security groups for EMR Notebooks (p. 541)
- Using Amazon EMR block public access (p. 542)

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. 307).

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 *inbound* rules that the new cluster needs in addition to rules that may have been added earlier. Unless specifically stated otherwise, each rule for default EMR-managed security groups is also added to custom EMR-managed security groups that you specify.

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. 534).
- **ElasticMapReduce-slave**

  For rules in this security group, see Amazon EMR-managed security group for core and task instances (public subnets) (p. 536).
- **ElasticMapReduce-Master-Private**

  For rules in this security group, see Amazon EMR-managed security group for the master instance (private subnets) (p. 536).
- **ElasticMapReduce-Slave-Private**

  For rules in this security group, see Amazon EMR-managed security group for core and task instances (private subnets) (p. 537).
- **ElasticMapReduce-ServiceAccess**

  For rules in this security group, see Amazon EMR-managed security group for service access (private subnets) (p. 539).

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**. It has the following rules. If you specify a custom managed security group, Amazon EMR will add all the same rules to your custom security group.

Type	Protocc	Port range	Source	Details
Inbound rules				
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 `ElasticMapReduce-master` 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		

Type	Protocc	Port range	Source	Details
Custom	TCP	8443	Various Amazon IP address ranges	These rules allow the cluster manager to communicate with the master node.

To allow SSH access for trusted sources for the ElasticMapReduce-master security group with the old console

To edit your security groups, you must have permission to manage security groups for the VPC that the cluster is in. For more information, see Changing Permissions for an IAM User and the Example Policy that allows managing EC2 security groups in the *IAM User Guide*.

1. Open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Choose **Clusters**. Choose the **Name** of the cluster you want to modify.
3. Choose the **Security groups for Master** link under **Security and access**.
4. Choose **ElasticMapReduce-master** from the list.
5. Choose the **Inbound rules** tab and then **Edit inbound rules**.
6. Check for an inbound rule that allows public access with the following settings. If it exists, choose **Delete** to remove it.

   - **Type**

     SSH
   - **Port**

     22
   - **Source**

     Custom 0.0.0.0/0

     Warning
     Before December 2020, the ElasticMapReduce-master security group had a pre-configured rule to allow inbound traffic on Port 22 from all sources. This rule was created to simplify initial SSH connections to the master node. We strongly recommend that you remove this inbound rule and restrict traffic to trusted sources.
7. Scroll to the bottom of the list of rules and choose **Add Rule**.
8. For **Type**, select **SSH**.

   Selecting SSH automatically enters **TCP** for **Protocol** and **22** for **Port Range**.
9. For source, select **My IP** to automatically add your IP address as the source address. You can also add a range of **Custom** trusted client IP addresses, or create additional rules for other clients. Many network environments dynamically allocate IP addresses, so you might need to update your IP addresses for trusted clients in the future.
10. Choose **Save**.
11. Optionally, choose **ElasticMapReduce-slave** from the list and repeat the steps above to allow SSH client access to core and task nodes.

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	Protocc	Port range	Source	Details
Inbound rules				
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 `ElasticMapReduce-slave` 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		

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	Protocc	Port range	Source	Details
Inbound rules				
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	These rules allow all inbound ICMP traffic and traffic over any TCP or UDP port from any core

Type	Protocc	Port range	Source	Details
All TCP	TCP	All	group for core and task nodes.	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 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.
Outbound rules				
All traffic	All	All	0.0.0.0/0	Provides outbound access to the internet.
Custom TCP	TCP	9443	The Group ID of the managed security group for service access in a private subnet.	If the above "All traffic" default outbound rule is removed, this rule is a minimum requirement for EMR 5.30.0 and later. **Note** Amazon EMR does not add this rule when you use a custom managed security group.
Custom TCP	TCP	80 (http) or 443 (https)	The Group ID of the managed security group for service access in a private subnet.	If the above "All traffic" default outbound rule is removed, this rule is a minimum requirement for EMR 5.30.0 and later to connect to Amazon S3 over https. **Note** Amazon EMR does not add this rule when you use a custom managed security group.

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	Protocc	Port range	Source	Details
Inbound rules				
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 `ElasticMapReduce-slave` 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
All TCP	TCP	All		
All UDP	UDP	All		

Type	Protocc	Port range	Source	Details
				custom managed security groups to restrict cross-cluster access.
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.
Outbound rules				
All traffic	All	All	0.0.0.0/0	See Editing outbound rules (p. 538) below.
Custom TCP	TCP	80 (http) or 443 (https)	The Group ID of the managed security group for service access in a private subnet.	If the above "All traffic" default outbound rule is removed, this rule is a minimum requirement for EMR 5.30.0 and later to connect to Amazon S3 over https. **Note** Amazon EMR does not add this rule when you use a custom managed security group.

Editing outbound rules

By default, Amazon EMR creates this security group with outbound rules that allow all outbound traffic on all protocols and ports. Allowing all outbound traffic is selected because various Amazon EMR and customer applications that can run on Amazon EMR clusters may require different egress rules. EMR is not able to anticipate these specific settings when creating default security groups. You can scope down egress in your security groups to include only those rules that suit your use cases and security policies. At minimum, this security group requires the following outbound rules, but some applications might need additional egress.

Type	Protocc	Port range	Destination	Details
All TCP	TCP	All	pl-*xxxxxxxx*	Managed Amazon S3 prefix list `com.amazonaws.`*`MyRegion`*`.s3`.
All Traffic	All	All	sg-*xxxxxxxxxxxxxxxxx*	The ID of the `ElasticMapReduce-Slave-Private` security group.
All Traffic	All	All	sg-*xxxxxxxxxxxxxxxxx*	The ID of the `ElasticMapReduce-Master-Private` security group.
Custom TCP	TCP	9443	sg-*xxxxxxxxxxxxxxxxx*	The ID of the `ElasticMapReduce-ServiceAccess` security group.

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 inbound rules, and outbound rules that allow traffic over HTTPS (port 8443, port 9443) to the other managed security groups in private subnets. These rules allow the cluster manager to communicate with the master node and with core and task nodes. The same rules are needed if you are using custom security groups.

Type	Protocc	Port range	Source	Details
Inbound rules Required for EMR clusters with Amazon EMR release 5.30.0 and later.				
Custom TCP	TCP	9443	The Group ID of the managed security group for master instance.	This rule allows the communication between master instance's security group to the service access security group.
Outbound rules Required for all EMR clusters				
Custom TCP	TCP	8443	The Group ID of the managed security group for master instance.	These rules allow the cluster manager to communicate with the master node and with core and task nodes.
Custom TCP	TCP	8443	The Group ID of the managed security group for core and task instances.	These rules allow the cluster manager to communicate with the master node and with core and task nodes.

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/emr.
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
`EmrManagedMasterSecurityGroup`	Use this parameter to specify a custom managed security group for the master instance. If this parameter is specified, `EmrManagedSlaveSecurityGroup` must also be specified. For clusters in private subnets, `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` 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. The security group you specify as `ServiceAccessSecurityGroup` should not be used for any other purpose and should also be reserved for Amazon EMR. If this parameter is

Parameter	Description
	specified, `EmrManagedMasterSecurityGroup` 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-5.36.0 --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-xxxxxxxxxxx,\
AdditionalMasterSecurityGroups=['sg-xxxxxxxxxxx',\
'sg-xxxxxxxxxxx','sg-xxxxxxxxxx'],\
AdditionalSlaveSecurityGroups=sg-xxxxxxxxxxx \
--instance-type m5.xlarge
```

For more information, see create-cluster in the *AWS CLI Command Reference*.

Specifying EC2 security groups for EMR Notebooks

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 (p. 541).

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

Default EC2 security group for EMR Notebooks

The default EC2 security group for the EMR notebook 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.

Custom EC2 security group for EMR Notebooks when associating Notebooks with Git repositories

To link a Git repository to your notebook, the security group for the EMR notebook must include an outbound rule to allow the notebook to route traffic to the internet. It is recommended that you create a new security group for this purpose. Updating the default **ElasticMapReduceEditors-Editor** security group may give the same outbound rules to other notebooks that are attached to this security group.

Rules

- Inbound

  None
- Outbound

  Allow the notebook to route traffic to the internet via the cluster, as the following example demonstrates. The value 0.0.0.0/0 is used for example purposes. You can modify this rule to specify the IP address(es) for your Git-based repositories.

Type	Protocol	Port range	Destination
Custom TCP rule	TCP	18888	SG-
HTTPS	**TCP**	**443**	**0.0.0.0/0**

Using Amazon EMR block public access

Amazon EMR block public access prevents a cluster in a public subnet from launching when any security group associated with the cluster has a rule that allows inbound traffic from IPv4 0.0.0.0/0 or IPv6 ::/0

(public access) on a port, unless the port has been specified as an exception. Port 22 is an exception by default. You can configure exceptions to allow public access on a port or range of ports. Block public access does not take effect in private subnets.

Block public access is enabled for each AWS Region for your AWS account. In other words, each Region has block public access enabled for all clusters created by your account in that Region.

> **Important**
> We do not recommend turning off block public access for your account.

Block public access is only applicable during cluster creation. Block public access does not block IAM principals with appropriate permissions from updating security group configurations to allow public access on running clusters.

Block public access is an account level configuration that helps you centrally manage public network access to Amazon EMR clusters in a Region. When account users launch clusters in the Region where you have enabled block public access configuration, Amazon EMR checks the port rules defined in the configuration and compare it with inbound traffic rules specified in the security groups associated with the clusters. If these security groups have inbound rules that open ports to the public IP address but these ports are not configured as exceptions in block public access configuration, Amazon EMR will fail the cluster creation and send an exception to the user.

You can update security groups and the block public access configuration in your accounts at any time. Amazon EMR does not check your clusters that are already running against block public access configuration after such updates and will not fail clusters that are already running.

Amazon EMR block public access is available in the following Regions:

- Africa (Cape Town) - af-south-1
- Asia Pacific (Mumbai) - ap-south-1
- Asia Pacific (Hong Kong) - ap-east-1
- Asia Pacific (Jakarta) - ap-southeast-3
- Asia Pacific (Osaka) - ap-northeast-3
- Asia Pacific (Seoul) - ap-northeast-2
- Asia Pacific (Singapore) - ap-southeast-1
- Asia Pacific (Sydney) - ap-southeast-2
- Asia Pacific (Tokyo) - ap-northeast-1
- Canada (Central) - ca-central-1
- China (Beijing) - cn-north-1
- China (Ningxia) - cn-northwest-1
- Europe (Frankfurt) - eu-central-1
- Europe (Ireland) - eu-west-1
- Europe (London) - eu-west-2
- Europe (Milano) - eu-south-1
- Europe (Paris) - eu-west-3
- Europe (Stockholm) - eu-north-1
- Middle East (Bahrain) - me-south-1
- South America (São Paulo) - sa-east-1
- US East (N. Virginia) - us-east-1
- US East (Ohio) - us-east-2
- US West (N. California) - us-west-1

- US West (Oregon) - us-west-2

Configure block public access

You can enable and disable block public access settings using the AWS Management Console, the AWS CLI, and the Amazon EMR API. Settings apply across your account on a Region-by-Region basis. To maintain cluster security, it's recommended that you keep BPA enabled.

To configure block public access using the AWS Management Console

1. Open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. On the navigation bar, make sure that the **Region** you want to configure is selected.
3. Choose **Block public access**.
4. Under **Block public access settings**, complete the following steps.

To...	Do this...
Turn block public access on or off	Choose **Change**, choose **On** or **Off** as appropriate, and then choose the check mark to confirm.
Edit ports in the list of exceptions	1. Under **Exceptions**, choose **Edit**. 2. To add ports to the list of exceptions, choose **Add a port range** and enter a new port or port range. Repeat for each port or port range to add. 3. To remove a port or port range, choose the **x** next to the entry in the **Port ranges** list. 4. Choose **Save Changes**.

To configure block public access using the AWS CLI

Use the `aws emr put-block-public-access-configuration` command to configure block public access as shown in the following examples.

To...	Do this...
Turn block public access on	Set `BlockPublicSecurityGroupRules` to `true` as shown in the following example. For the cluster to launch, no security group associated with a cluster can have an inbound rule that allows public access. `aws emr put-block-public-access-configuration --block-public-access-configuration BlockPublicSecurityGroupRules=true`
Turn block public access off	Set `BlockPublicSecurityGroupRules` to `false` as shown in the following example. Security groups associated with a cluster can have inbound rules that allow public access on any port. We do not recommend this configuration.

To...	Do this...
	`aws emr put-block-public-access-configuration --block-public-access-configuration BlockPublicSecurityGroupRules=false`
Turn block public access on and specify ports as exceptions	The following example turns on block public access, and specifies Port 22 and Ports 100-101 as exceptions. This allows clusters to be created if an associated security group has an inbound rule that allows public access on Port 22, Port 100, or Port 101. `aws emr put-block-public-access-configuration --block-public-access-configuration '{ "BlockPublicSecurityGroupRules": true, "PermittedPublicSecurityGroupRuleRanges": [ { "MinRange": 22, "MaxRange": 22 }, { "MinRange": 100, "MaxRange": 101 } ] }'`

Compliance validation for Amazon EMR

Third-party auditors assess the security and compliance of Amazon EMR as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others.

For a list of AWS services in scope of specific compliance programs, see AWS services in scope by compliance program. For general information, see AWS compliance programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading reports in AWS Artifact.

Your compliance responsibility when using Amazon EMR is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. If your use of Amazon EMR is subject to compliance with standards such as HIPAA, PCI, or FedRAMP, AWS provides resources to help:

- Security and compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- Architecting for HIPAA Security and Compliance whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- AWS compliance resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Config – This AWS service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience in Amazon EMR

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency,

high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS global infrastructure.

In addition to the AWS global infrastructure, Amazon EMR offers several features to help support your data resiliency and backup needs.

- **Integration with Amazon S3 through EMRFS**
- **Support for multiple master nodes**

Infrastructure security in Amazon EMR

As a managed service, Amazon EMR is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of security processes whitepaper.

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

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

Topics

- Connect to Amazon EMR using an interface VPC endpoint (p. 546)

Connect to Amazon EMR using an interface VPC endpoint

You can connect directly to Amazon EMR using an interface VPC endpoint (AWS PrivateLink) in your Virtual Private Cloud (VPC) instead of connecting over the internet. When you use an interface VPC endpoint, communication between your VPC and Amazon EMR is conducted entirely within the AWS network. Each VPC endpoint is represented by one or more Elastic network interfaces (ENIs) with private IP addresses in your VPC subnets.

The interface VPC endpoint connects your VPC directly to Amazon EMR without an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. The instances in your VPC don't need public IP addresses to communicate with the Amazon EMR API.

To use Amazon EMR through your VPC, you must connect from an instance that is inside the VPC or connect your private network to your VPC by using an Amazon Virtual Private Network (VPN) or AWS Direct Connect. For information about Amazon VPN, see VPN connections in the *Amazon Virtual Private Cloud User Guide*. For information about AWS Direct Connect, see Creating a connection in the *AWS Direct Connect User Guide*.

You can create an interface VPC endpoint to connect to Amazon EMR using the AWS console or AWS Command Line Interface (AWS CLI) commands. For more information, see Creating an interface endpoint.

After you create an interface VPC endpoint, if you enable private DNS hostnames for the endpoint, the default Amazon EMR endpoint resolves to your VPC endpoint. The default service name endpoint for Amazon EMR is in the following format.

```
elasticmapreduce.Region.amazonaws.com
```

If you do not enable private DNS hostnames, Amazon VPC provides a DNS endpoint name that you can use in the following format.

```
VPC_Endpoint_ID.elasticmapreduce.Region.vpce.amazonaws.com
```

For more information, see Interface VPC endpoints (AWS PrivateLink) in the *Amazon VPC User Guide*.

Amazon EMR supports making calls to all of its API actions inside your VPC.

You can attach VPC endpoint policies to a VPC endpoint to control access for IAM principals. You can also associate security groups with a VPC endpoint to control inbound and outbound access based on the origin and destination of network traffic, such as a range of IP addresses. For more information, see Controlling access to services with VPC endpoints.

Create a VPC endpoint policy for Amazon EMR

You can create a policy for Amazon VPC endpoints for Amazon EMR to specify the following:

- The principal that can or cannot perform actions
- The actions that can be performed
- The resources on which actions can be performed

For more information, see Controlling access to services with VPC endpoints in the *Amazon VPC User Guide*.

Example – VPC endpoint policy to deny all access from a specified AWS account

The following VPC endpoint policy denies AWS account *123456789012* all access to resources using the endpoint.

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "*",
            "Effect": "Deny",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

Example – VPC endpoint policy to allow VPC access only to a specified IAM principal (user)

The following VPC endpoint policy allows full access only to the IAM user *lijuan* in AWS account *123456789012*. All other IAM principals are denied access using the endpoint.

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:user/lijuan"
                ]
            }
        }]
}
```

Example – VPC endpoint policy to allow read-only EMR operations

The following VPC endpoint policy allows only AWS account *123456789012* to perform the specified Amazon EMR actions.

The actions specified provide the equivalent of read-only access for Amazon EMR. All other actions on the VPC are denied for the specified account. All other accounts are denied any access. For a list of Amazon EMR actions, see Actions, resources, and condition keys for Amazon EMR.

```
{
    "Statement": [
        {
            "Action": [
                "elasticmapreduce:DescribeSecurityConfiguration",
                "elasticmapreduce:GetBlockPublicAccessConfiguration",
                "elasticmapreduce:ListBootstrapActions",
                "elasticmapreduce:ViewEventsFromAllClustersInConsole",
                "elasticmapreduce:ListSteps",
                "elasticmapreduce:ListInstanceFleets",
                "elasticmapreduce:DescribeCluster",
                "elasticmapreduce:ListInstanceGroups",
                "elasticmapreduce:DescribeStep",
                "elasticmapreduce:ListInstances",
                "elasticmapreduce:ListSecurityConfigurations",
                "elasticmapreduce:DescribeEditor",
                "elasticmapreduce:ListClusters",
                "elasticmapreduce:ListEditors"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

Example – VPC endpoint policy denying access to a specified cluster

The following VPC endpoint policy allows full access for all accounts and principals, but denies any access for AWS account *123456789012* to actions performed on the Amazon EMR cluster with cluster ID

j-A1B2CD34EF5G. Other Amazon EMR actions that don't support resource-level permissions for clusters are still allowed. For a list of Amazon EMR actions and their corresponding resource type, see Actions, resources, and condition keys for Amazon EMR.

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "*",
            "Effect": "Deny",
            "Resource": "arn:aws:elasticmapreduce:us-west-2:123456789012:cluster/j-
A1B2CD34EF5G",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

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. 550)
- Connect to the cluster (p. 591)
- Terminate a cluster (p. 607)
- Scaling cluster resources (p. 610)
- Cloning a cluster using the console (p. 641)
- Submit work to a cluster (p. 642)
- Automate recurring clusters with AWS Data Pipeline (p. 647)

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.

Application history is also available from the console using the "persistent" application UIs for Spark History Server starting with Amazon EMR 5.25.0. With Amazon EMR 6.x, persistent YARN timeline server, and Tez user interfaces are also available. These services are hosted off-cluster, so you can access application history for 30 days after the cluster terminates, without the need for a SSH connection or web proxy. See View application history.

Topics

- View cluster status and details (p. 550)
- Enhanced step debugging (p. 556)
- View application history (p. 557)
- View log files (p. 564)
- View cluster instances in Amazon EC2 (p. 568)
- CloudWatch events and metrics (p. 568)
- View cluster application metrics with Ganglia (p. 589)
- Logging Amazon EMR API calls in AWS CloudTrail (p. 589)

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. 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.

You can access application history stored on-cluster for one week from the time it is recorded, regardless of whether the cluster is running or terminated. In addition, persistent application user interfaces store application history off-cluster for 30 days after a cluster terminates. See View application history.

For more information about cluster states, such as Waiting and Running, see Understanding the cluster lifecycle (p. 3).

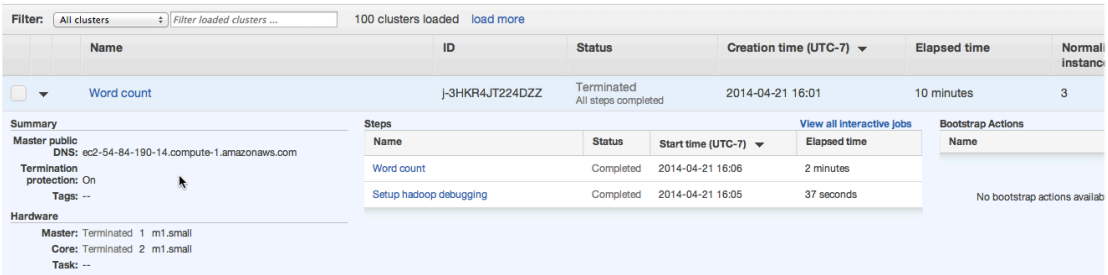
View cluster status using the AWS Management Console

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**.

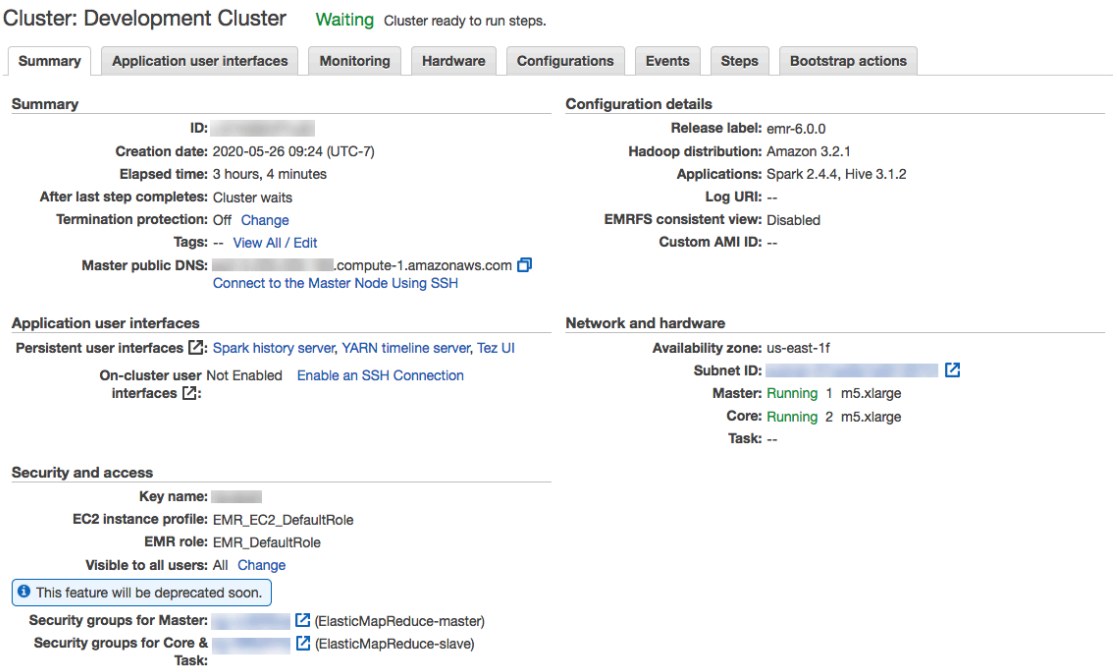
  The cluster's row expands to provide more information about the cluster, hardware, steps, and bootstrap actions. Use the links in this section to drill into specifics. For example, click a link under **Steps** to access step log files, see the JAR associated with the step, drill into the step's jobs and tasks, and access log files.



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:



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 user interfaces	Use this tab to view persistent off-cluster YARN timeline server and Tez UI 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. 557). On-cluster application user interfaces are available while the cluster is running.
Monitoring	Use this tab to view graphs depicting key indicators of cluster operation over a time 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. 569).

Tab	Information
Steps	Use this tab to see the status and access log files for steps that you submitted. For more information about steps, see Submit work to a cluster (p. 642).
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 *Amazon EMR Release Guide*.
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. 191).

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. For more information about cluster states, see Understanding the cluster lifecycle (p. 3). 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-cXXXXX0",
            "IamInstanceProfile": "EMR_EC2_DefaultRole",
            "Ec2KeyName": "myKey",
            "Ec2AvailabilityZone": "us-east-1c",
            "EmrManagedSlaveSecurityGroup": "sg-example"
```

```
        },
        "Name": "Development Cluster",
        "ServiceRole": "EMR_DefaultRole",
        "Tags": [],
        "TerminationProtected": false,
        "ReleaseLabel": "emr-4.0.0",
        "NormalizedInstanceHours": 16,
        "InstanceGroups": [
            {
                "RequestedInstanceCount": 1,
                "Status": {
                    "Timeline": {
                        "ReadyDateTime": 1438281058.101,
                        "CreationDateTime": 1438280702.499
                    },
                    "State": "RUNNING",
                    "StateChangeReason": {
                        "Message": ""
                    }
                },
                "Name": "CORE",
                "InstanceGroupType": "CORE",
                "Id": "ig-2EEXAMPLEXXP",
                "Configurations": [],
                "InstanceType": "m5.xlarge",
                "Market": "ON_DEMAND",
                "RunningInstanceCount": 1
            },
            {
                "RequestedInstanceCount": 1,
                "Status": {
                    "Timeline": {
                        "ReadyDateTime": 1438281023.879,
                        "CreationDateTime": 1438280702.499
                    },
                    "State": "RUNNING",
                    "StateChangeReason": {
                        "Message": ""
                    }
                },
                "Name": "MASTER",
                "InstanceGroupType": "MASTER",
                "Id": "ig-2A1234567XP",
                "Configurations": [],
                "InstanceType": "m5.xlarge",
                "Market": "ON_DEMAND",
                "RunningInstanceCount": 1
            }
        ],
        "Applications": [
            {
                "Version": "1.0.0",
                "Name": "Hive"
            },
            {
                "Version": "2.6.0",
                "Name": "Hadoop"
            },
            {
                "Version": "0.14.0",
                "Name": "Pig"
            },
            {
                "Version": "1.4.1",
                "Name": "Spark"
            }
```

```
        ],
        "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 data range, use the `list-clusters` command with the `--created-after` and `--created-before` parameters.

The following command lists all clusters created between October 09, 2019 and October 12, 2019.

```
aws emr list-clusters --created-after 2019-10-09T00:12:00 --created-
before 2019-10-12T00:12:00
```

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
```

For more information about cluster states, see Understanding the cluster lifecycle (p. 3).

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:

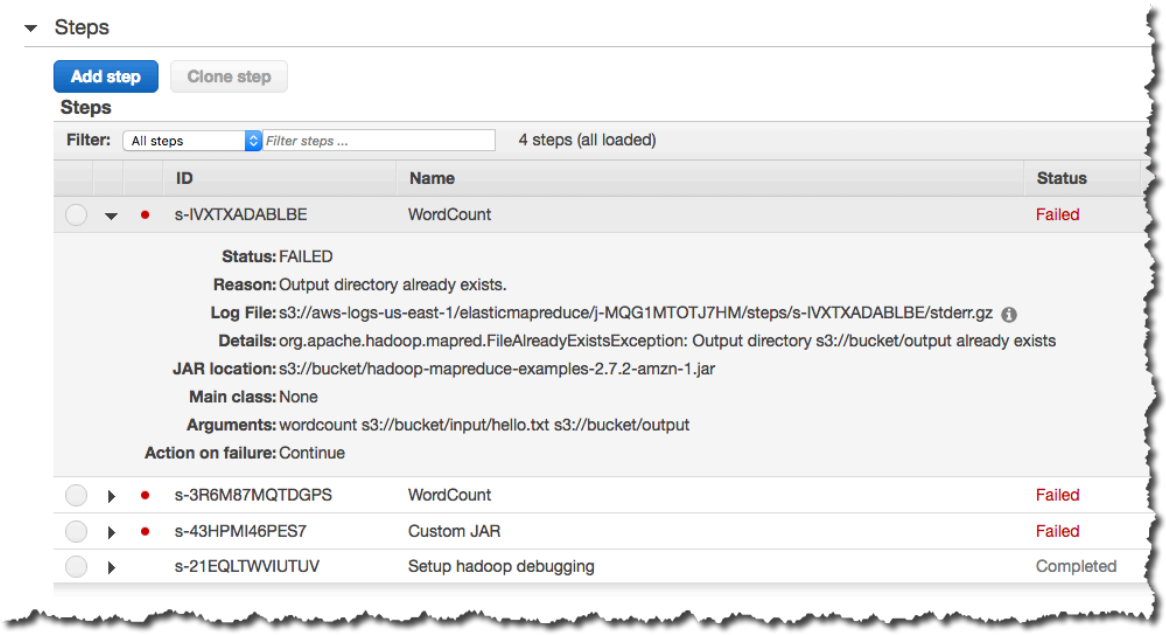
- A common Hadoop error such as the output directory already exists, the input directory does not exist, or an application runs out of memory.
- Java errors such as an application that was compiled with an incompatible version of Java or run with a main class that is not found.
- An issue accessing objects 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/emr.
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.



To view failure details using the AWS CLI

- To get failure details for a step using the AWS CLI, use the describe-step command.

```
aws emr describe-step --cluster-id j-1K48XXXXXHCB --step-id s-3QM0XXXXXM1W
```

The output will look similar to the following:

```
{
  "Step": {
    "Status": {
      "FailureDetails": {
        "LogFile": "s3://myBucket/logs/j-1K48XXXXXHCB/steps/s-3QM0XXXXXM1W/stderr.gz",
        "Message": "org.apache.hadoop.mapred.FileAlreadyExistsException: Output
 directory s3://myBucket/logs/beta already exists",
        "Reason": "Output directory already exists."
      },
      "Timeline": {
        "EndDateTime": 1469034209.143,
        "CreationDateTime": 1469033847.105,
        "StartDateTime": 1469034202.881
      },
      "State": "FAILED",
      "StateChangeReason": {}
    },
    "Config": {
      "Args": [
        "wordcount",
        "s3://myBucket/input/input.txt",
        "s3://myBucket/logs/beta"
      ],
      "Jar": "s3://myBucket/jars/hadoop-mapreduce-examples-2.7.2-amzn-1.jar",
      "Properties": {}
    },
    "Id": "s-3QM0XXXXXM1W",
    "ActionOnFailure": "CONTINUE",
    "Name": "ExampleJob"
  }
}
```

View application history

You can view Spark History Server and YARN timeline service application details using the **Application user interfaces** tab of a cluster's detail page in the console. Amazon EMR application history makes it easier for you to troubleshoot and analyze active jobs and job history.

The **Application user interfaces** tab provides several viewing options:

- Off-cluster access to persistent application user interfaces – Starting with Amazon EMR version 5.25.0, persistent application user interface links are available for Spark. With Amazon EMR version 5.30.1 and later, Tez UI and the YARN timeline server also have persistent application user interfaces. The YARN timeline server and Tez UI are open-source applications that provide metrics for active and terminated clusters. The Spark user interface provides details about scheduler stages and tasks, RDD sizes and memory usage, environmental information, and information about the running executors. Persistent application UIs are run off-cluster, so cluster information and logs are available for 30 days after an application terminates. Unlike on-cluster application user interfaces, persistent application UIs don't require you to set up a web proxy through a SSH connection.
- On-cluster application user interfaces – There are a variety of application history user interfaces that can be run on a cluster. On-cluster user interfaces are hosted on the master node and require you to set up a SSH connection to the web server. On-cluster application user interfaces keep application

history for one week after an application terminates. For more information and instructions on setting up an SSH tunnel, see View web interfaces hosted on Amazon EMR clusters (p. 599).

With the exception of the Spark History Server, YARN timeline server, and Hive applications, on-cluster application history can only be viewed while the cluster is running.

- High-level application history – With Amazon EMR version 5.8.0 to 5.36.0 and 6.x releases up to 6.8.0, you can view a summary of application history in the EMR console, including key metrics for stage tasks and executors. The summary of the application history is available for all YARN applications. Additional details are provided for Spark applications, but these details are only a subset of the information available through the Spark application UI.

View persistent application user interfaces

Starting with Amazon EMR version 5.25.0, you can connect to the persistent Spark History Server application details hosted off-cluster using the cluster **Summary** page or the **Application user interfaces** tab in the console. Tez UI and YARN timeline server persistent application interfaces are available starting with Amazon EMR version 5.30.1. One-click link access to persistent application history provides the following benefits:

- You can quickly analyze and troubleshoot active jobs and job history without setting up a web proxy through an SSH connection.
- You can access application history and relevant log files for active and terminated clusters. The logs are available for 30 days after the application ends.

On the **Application user interfaces** tab or the cluster **Summary** page for your cluster in the Amazon EMR 5.30.1 or 6.x console, choose the **YARN timeline server**, **Tez UI**, or **Spark history server** link.

The Application UI opens in a new browser tab. For more information, see Monitoring and instrumentation.

You can view YARN container logs through the links on the Spark history server, YARN timeline server, and Tez UI.

> **Note**
> To access YARN container logs from the Spark history server, YARN timeline server, and Tez UI, you must enable logging to Amazon S3 for your cluster. If logging is not enabled, the links to YARN container logs will not work.

Logs collection

To enable one-click access to persistent application user interfaces, Amazon EMR collects two types of logs:

- **Application event logs** are collected into an EMR system bucket. The event logs are encrypted at rest using Server-Side Encryption with Amazon S3 Managed Keys (SSE-S3). If you use a private subnet for your cluster, make sure to include `"arn:aws:s3:::prod.MyRegion.appinfo.src/*"` in the

resource list of the Amazon S3 policy for the private subnet. For more information, see Minimum Amazon S3 policy for private subnet.

- **YARN container logs** are collected into an Amazon S3 bucket that you own. You must enable logging for your cluster to access YARN container logs. For more information, see Configure cluster logging and debugging.

If you need to disable this feature for privacy reasons, you can stop the daemon by using a bootstrap script when you create a cluster, as the following example demonstrates.

```
aws emr create-cluster --name "Stop Application UI Support" --release-label emr-5.36.0 \
--applications Name=Hadoop Name=Spark --ec2-attributes KeyName=<myEMRKeyPairName> \
--instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m3.xlarge
 InstanceGroupType=CORE,InstanceCount=1,InstanceType=m3.xlarge
 InstanceGroupType=TASK,InstanceCount=1,InstanceType=m3.xlarge \
--use-default-roles --bootstrap-actions Path=s3://region.elasticmapreduce/bootstrap-
actions/run-if,Args=["instance.isMaster=true","echo Stop Application UI | sudo tee /etc/
apppusher/run-apppusher; sudo systemctl stop apppusher || exit 0"]
```

After you run this bootstrap script, Amazon EMR will not collect any Spark History Server or YARN timeline server event logs into the EMR system bucket. No application history information will be available on the **Application user interfaces** tab, and you will lose access to all application user interfaces from the console.

Large Spark event log files

In some cases, long-running Spark jobs, such as Spark streaming, and large jobs, such as Spark SQL queries, can generate large event logs. With large events logs, you can quickly use up disk space on compute instances and encounter `OutOfMemory` errors when you load Persistent UIs. To avoid these issues, we recommend that you turn on the Spark event log rolling and compaction feature. This feature is available on Amazon EMR versions emr-6.1.0 and later. For more details about rolling and compaction, see Applying compaction on rolling event log files in the Spark documentation.

To activate the Spark event log rolling and compaction feature, turn on the following Spark configuration settings.

- `spark.eventLog.rolling.enabled` – Turns on event log rolling based on size. This setting is deactivated by default.
- `spark.eventLog.rolling.maxFileSize` – When rolling is activated, specifies the maximum size of the event log file before it rolls over. The default is 128 MB.
- `spark.history.fs.eventLog.rolling.maxFilesToRetain` – Specifies the maximum number of non-compacted event log files to retain. By default, all event log files are retained. Set to a lower number to compact older event logs. The lowest value is 1.

Note that compaction attempts to exclude events with outdated event log files, such as the following. If it does discard events, you no longer see them on the Spark History Server UI.

- Events for finished jobs and related stage or task events.
- Events for terminated executors.
- Events for completed SQL inquiries, and related job, stage, and tasks events.

To launch a cluster with rolling and compaction enabled

1. Create a `spark-configuration.json` file with the following configuration.

```
[
```

```
   {
     "Classification": "spark-defaults",
        "Properties": {
           "spark.eventLog.rolling.enabled": true,
           "spark.history.fs.eventLog.rolling.maxFilesToRetain": 1
        }
   }
]
```

2. Create your cluster with the Spark rolling compaction configuration as follows.

```
aws emr create-cluster \
--release-label emr-6.6.0 \
--instance-type m4.large \
--instance-count 2 \
--use-default-roles \
--configurations file://spark-configuration.json
```

Considerations and limitations

One-click access to persistent application user interfaces currently has the following limitations.

- There will be at least a two-minute delay when the application details show up on the Spark History Server UI.
- This feature works only when the event log directory for the application is in HDFS. By default, Amazon EMR stores event logs in a directory of HDFS. If you change the default directory to a different file system, such as Amazon S3, this feature will not work.
- This feature is currently not available for EMR clusters with multiple master nodes or for EMR clusters integrated with AWS Lake Formation.
- To enable one-click access to persistent application user interfaces, you must have permission to the `DescribeCluster` action for Amazon EMR. If you deny an IAM principal's permission to this action, it takes approximately five minutes for the permission change to propagate.
- If you reconfigure applications in a running cluster, the application history will be not available through the application UI.
- For each AWS account, the default limit for active application UIs is 200.
- You can access application UIs from the console in the US East (N. Virginia) Region, US West (N. California) Region, Canada (Central) Region, EU (Frankfurt, Ireland, London, Paris, Stockholm), Asia Pacific (Mumbai, Seoul, Singapore, Sydney, and Tokyo), South America (São Paulo), China (Beijing) operated by Sinnet, and China (Ningxia) operated by NWCD.

View a high-level application history

With some Amazon EMR versions, you can view a high-level application history from the **Application user interfaces** tab in the Amazon EMR console. Amazon EMR keeps the summary of application history for seven days after an application has completed.

Considerations and limitations

Consider the following limitations when you use the **Application user interfaces** tab in the Amazon EMR console.

- You can only access the high-level application history feature when using Amazon EMR releases 5.8.0 to 5.36.0 and 6.x releases up to 6.8.0.
- The high-level application history feature does not support Spark Streaming applications.

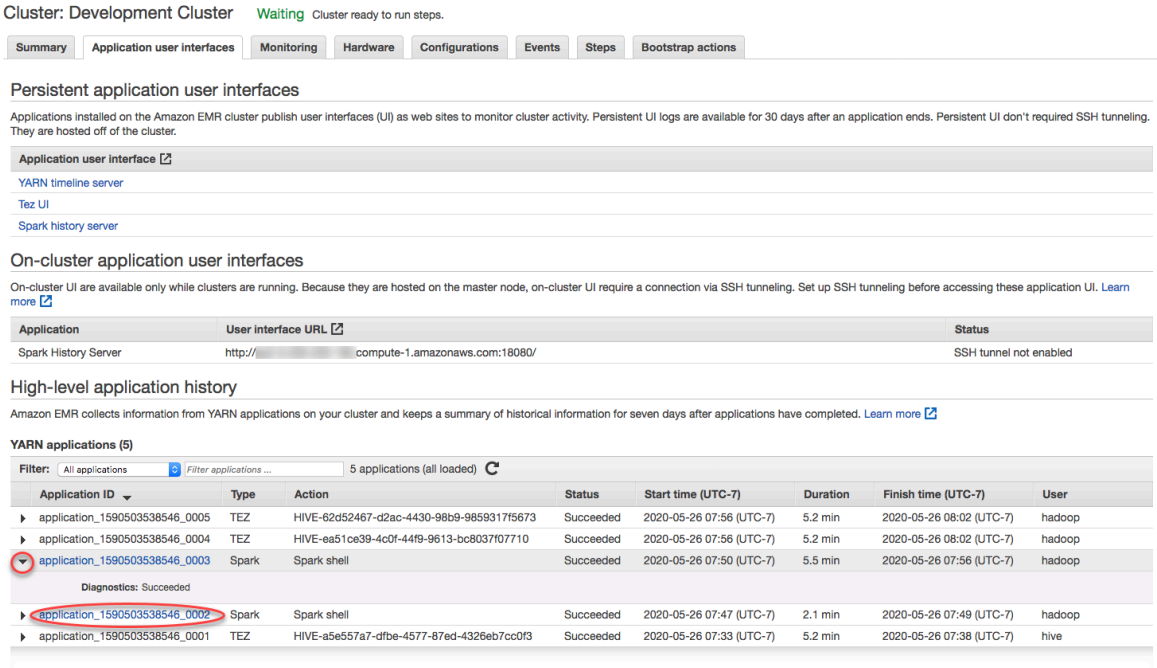
- One-click access to persistent application user interfaces is currently not available for Amazon EMR clusters with multiple master nodes or for Amazon EMR clusters integrated with AWS Lake Formation.

Example: View a high-level application history

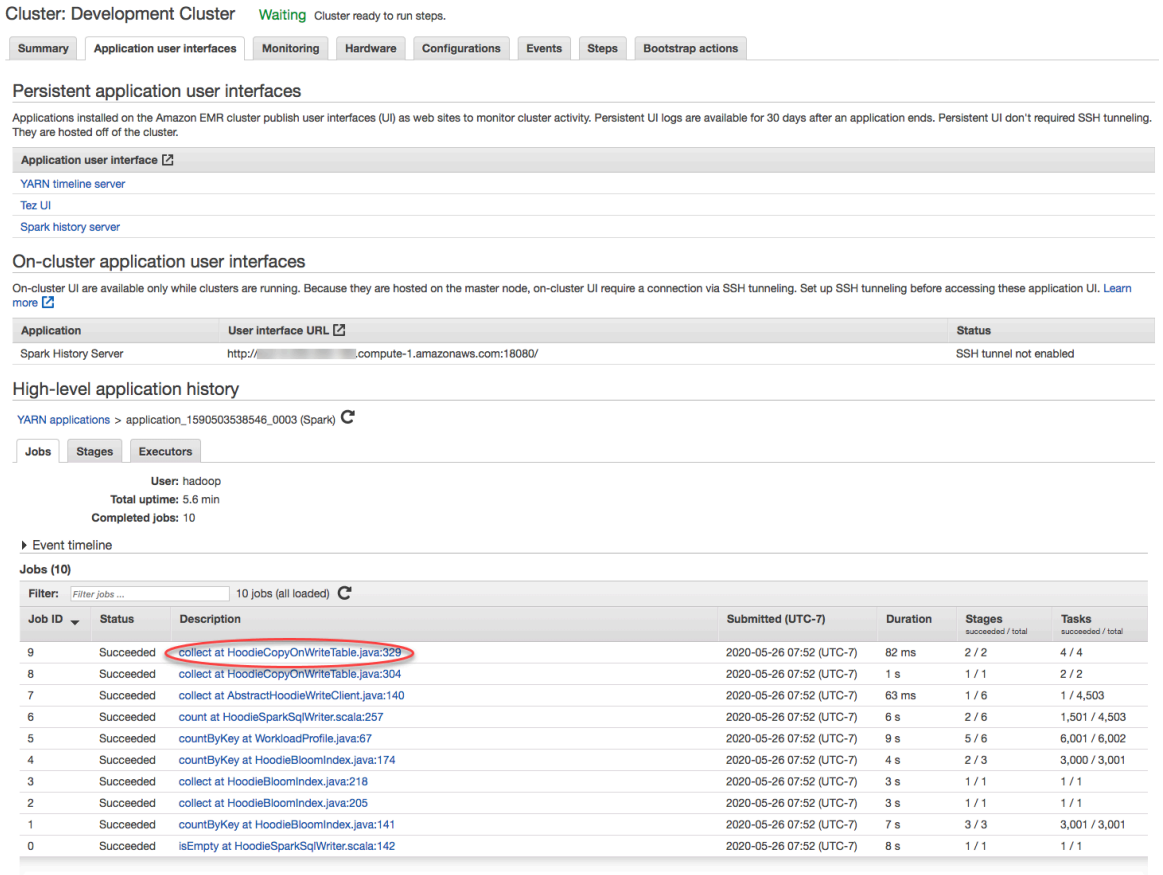
The following sequence demonstrates a drill-down through a Spark or YARN application into job details using the **Application user interfaces** tab on the cluster details page.

To view cluster details, select a cluster **Name** from the **Clusters** list. To view information about YARN container logs, you must enable logging for your cluster. For more information, see Configure cluster logging and debugging. For Spark application history, the information provided in the summary table is only a subset of the information available through the Spark history server UI.

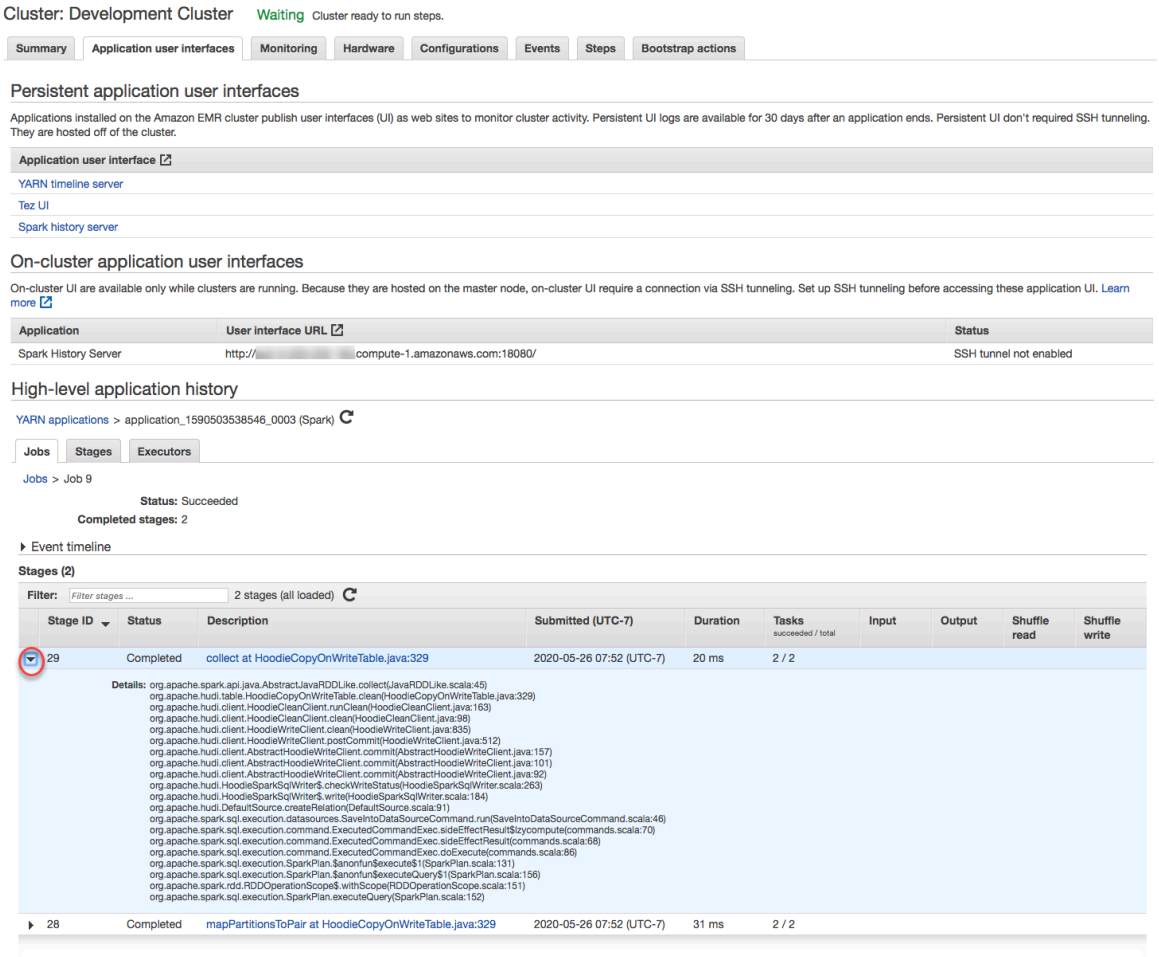
In the **Application user interfaces** tab under **High-level application history**, you can expand a row to show the diagnostic summary for a Spark application or select an **Application ID** link to view details about a different application.



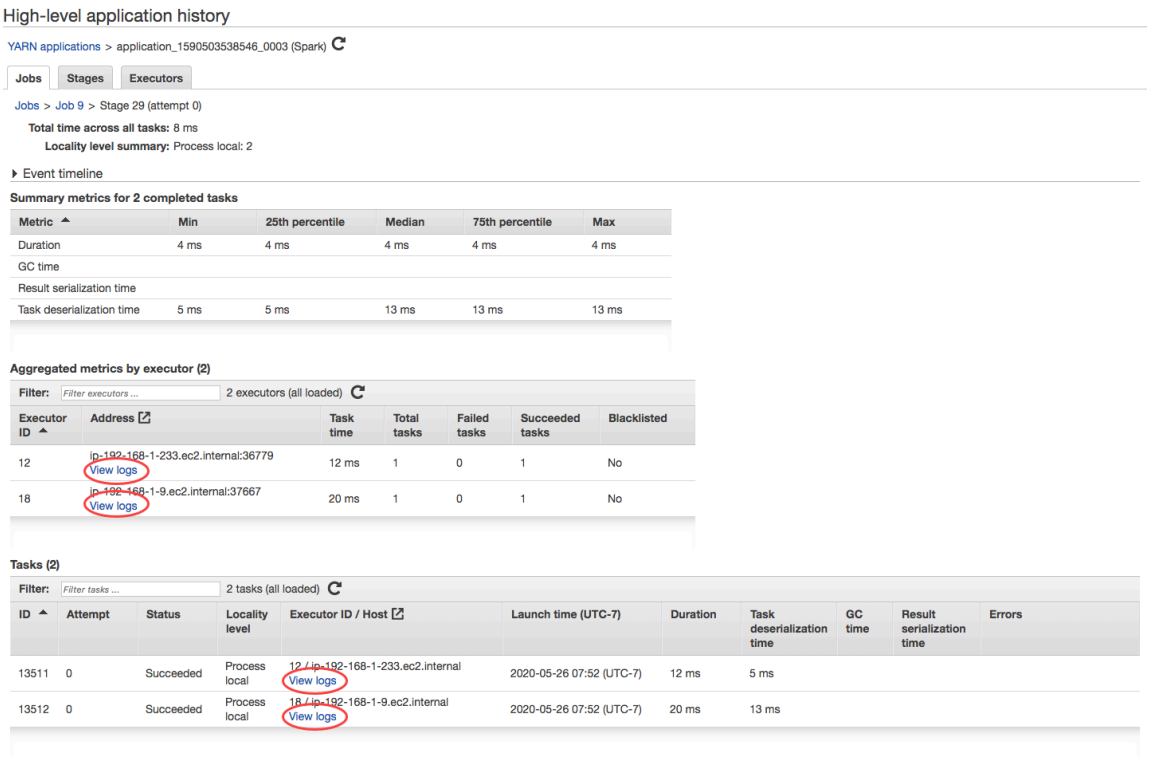
When you select an **Application ID** link, the UI changes to show the **YARN application** details for that application. In the **Jobs** tab of **YARN application** details, you can choose the **Description** link for a job to display details for that job.



On the job details page, you can expand information about individual job stages, and then select the **Description** link to see stage details.



On the stage details page, you can view key metrics for stage tasks and executors. You can also view task and executor logs using the **View logs** links.



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. 564)
- View log files archived to Amazon S3 (p. 565)
- View log files in the debugging tool (p. 567)

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
/emr/instance-controller/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.

Location	Description
/emr/instance-controller/log	Instance controller logs.
/emr/instance-state	Instance state logs. These contain information about the CPU, memory state, and garbage collector threads of the node.
/emr/service-nanny	Logs written by the service nanny process.
/mnt/var/log/*application*	Logs specific to an application such as Hadoop, Spark, or Hive.
/mnt/var/log/hadoop/steps/*N*	Step logs that contain information about the processing of the step. The value of *N* indicates the stepId assigned by Amazon EMR. For example, a cluster has two steps: `s-1234ABCDEFGH` and `s-5678IJKLMNOP`. The first step is located in `/mnt/var/log/hadoop/steps/s-1234ABCDEFGH/` and the second step in `/mnt/var/log/hadoop/steps/s-5678IJKLMNOP/`. The step logs written by Amazon EMR are as follows. • **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. 593).
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 /*cluster-id*/ folder, where *cluster-id* is the cluster ID.

The following table lists some of the log files you'll find on Amazon S3.

Location	Description
/*cluster-id*/node/	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.
/*cluster-id*/node/*instance-id*/*application*	The logs created by each application or daemon associated with an application. For example, the Hive server log is located at *cluster-id*/node/*instance-id*/hive/hive-server.log.
/*cluster-id*/steps/*step-id*/	Step logs that contain information about the processing of the step. The value of *step-id* indicates the step ID assigned by Amazon EMR. For example, a cluster has two steps: s-1234ABCDEFGH and s-5678IJKLMNOP. The first step is located in /mnt/var/log/hadoop/steps/s-1234ABCDEFGH/ and the second step in /mnt/var/log/hadoop/steps/s-5678IJKLMNOP/. The step logs written by Amazon EMR are as follows. • **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.
/*cluster-id*/containers	Application container logs. The logs for each YARN application are stored in these locations.
/*cluster-id*/hadoop-mapreduce/	The logs that contain information about configuration details and job history of MapReduce jobs.

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. Download the log file object to view it. For instructions, see Downloading an object.

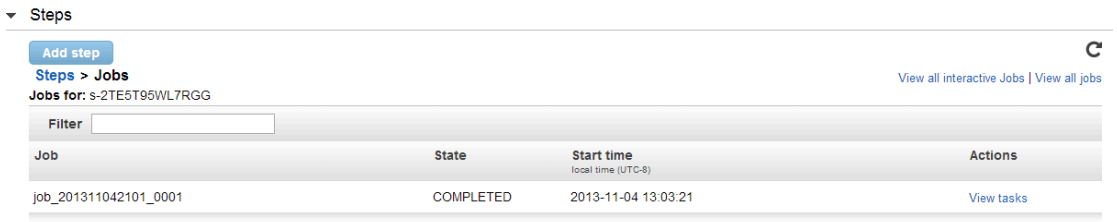
View log files in the debugging tool

Amazon EMR does not automatically enable the debugging tool. You must configure this when you launch the cluster.

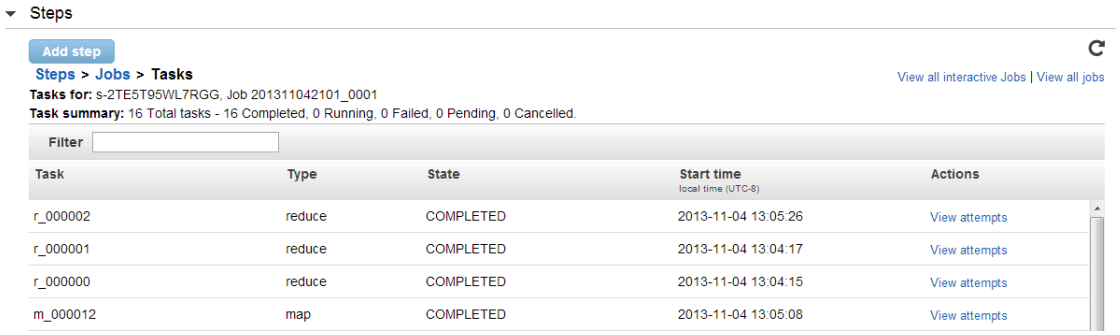
To view cluster logs using the console

1. Open the Amazon EMR console at https://console.aws.amazon.com/emr.
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-XXXXXXXXXXXXX.
- 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

Use events and metrics to track the activity and health of an Amazon EMR cluster. 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. 569)
- Monitor metrics with CloudWatch (p. 577)

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 details 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 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 *`ClusterId (ClusterName)`* was requested at *`Time`* and is being created.
`STARTING`	`INFO`	**Note** Applies only to clusters with the instance fleets configuration and multiple subnets selected within a VPC. Amazon EMR cluster *`ClusterId (ClusterName)`* is being created in subnet (*`SubnetName`*) in VPC (*`VPCName`*) in availability zone (*`AvailabilityZoneID`*), which was chosen from the specified VPC options.
`STARTING`	`INFO`	**Note** Applies only to clusters with the instance fleets configuration and multiple Availability Zones selected within EC2-Classic. Amazon EMR cluster *`ClusterId (ClusterName)`* is being created in availability zone (*`AvailabilityZoneID`*), which was chosen from the specified availability zone options.

State or state change	Severity	Message
RUNNING	INFO	Amazon EMR cluster *ClusterId (ClusterName)* began running steps at *Time*.
WAITING	INFO	Amazon EMR cluster *ClusterId (ClusterName)* was created at *Time* and is ready for use. - or - Amazon EMR cluster *ClusterId (ClusterName)* finished running all pending steps at *Time*. **Note** A cluster in the WAITING state may nevertheless be processing jobs.
TERMINATED	The severity depends on the reason for the state change, as shown in the following: • **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 *ClusterId (ClusterName)* has terminated at *Time* with a reason of *StateChangeReason:Code*.
TERMINATED_WITH_ERRORS	CRITICAL	Amazon EMR Cluster *ClusterId (ClusterName)* has terminated with errors at *Time* with a reason of *StateChangeReason:Code*.

Instance fleet events

Note
The instance fleets configuration is available only in Amazon EMR releases 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 *InstanceFleetID* in

State or state change	Severity	Message
		Amazon EMR cluster *ClusterId (ClusterName)* is complete. Provisioning started at *Time* and took *Num* minutes. The instance fleet now has On-Demand capacity of *Num* and Spot capacity of *Num*. Target On-Demand capacity was *Num*, and target Spot capacity was *Num*.
From `WAITING` to `RESIZING`	`INFO`	A resize for instance fleet *InstanceFleetID* in Amazon EMR cluster *ClusterId (ClusterName)* started at *Time*. The instance fleet is resizing from an On-Demand capacity of *Num* to a target of *Num*, and from a Spot capacity of *Num* to a target of *Num*.
From `RESIZING` to `WAITING`	`INFO`	The resizing operation for instance fleet *InstanceFleetID* in Amazon EMR cluster *ClusterId (ClusterName)* is complete. The resize started at *Time* and took *Num* minutes. The instance fleet now has On-Demand capacity of *Num* and Spot capacity of *Num*. Target On-Demand capacity was *Num* and target Spot capacity was *Num*.
From `RESIZING` to `WAITING`	`WARN`	The resizing operation for instance fleet *InstanceFleetID* in Amazon EMR cluster *ClusterId (ClusterName)* has reached the timeout and stopped. The resize started at *Time* and stopped after *Num* minutes. The instance fleet now has On-Demand capacity of *Num* and Spot capacity of *Num*. Target On-Demand capacity was *Num* and target Spot capacity was *Num*.
`SUSPENDED`	`ERROR`	Instance fleet *InstanceFleetID* in Amazon EMR cluster *ClusterId (ClusterName)* was arrested at *Time* for the following reason: *ReasonDesc*.

State or state change	Severity	Message
RESIZING	WARNING	The resizing operation for instance fleet *InstanceFleetID* in Amazon EMR cluster *ClusterId (ClusterName)* is stuck for the following reason: *ReasonDesc*.
WAITING or RUNNING	INFO	A resize for instance fleet *InstanceFleetID* in Amazon EMR cluster *ClusterId (ClusterName)* was initiated by *Entity* at *Time*.

Instance group events

State or state change	Severity	Message
From RESIZING to RUNNING	INFO	The resizing operation for instance group *InstanceGroupID* in Amazon EMR cluster *ClusterId (ClusterName)* is complete. It now has an instance count of *Num*. The resize started at *Time* and took *Num* minutes to complete.
From RUNNING to RESIZING	INFO	A resize for instance group *InstanceGroupID* in Amazon EMR cluster *ClusterId (ClusterName)* started at *Time*. It is resizing from an instance count of *Num* to *Num*.
SUSPENDED	ERROR	Instance group *InstanceGroupID* in Amazon EMR cluster *ClusterId (ClusterName)* was arrested at *Time* for the following reason: *ReasonDesc*.
RESIZING	WARNING	The resizing operation for instance group *InstanceGroupID* in Amazon EMR cluster *ClusterId (ClusterName)* is stuck for the following reason: *ReasonDesc*.
From RUNNING to RESIZING	INFO	A resize for instance group *InstanceGroupID* in Amazon EMR cluster *ClusterId (ClusterName)* was initiated by *Entity* at *Time*.

Note
With Amazon EMR version 5.21.0 and later, you can override cluster configurations and specify additional configuration classifications for each instance group in a running cluster. You do this by using the Amazon EMR console, the AWS Command Line Interface (AWS CLI), or the AWS SDK. For more information, see Supplying a Configuration for an Instance Group in a Running Cluster.

The following table lists Amazon EMR events for the reconfiguration operation, along with the state or state change that the event indicates, the severity of the event, and event messages.

State or state change	Severity	Message
RUNNING	INFO	A reconfiguration for instance group *InstanceGroupID* in the Amazon EMR cluster *ClusterId (ClusterName)* was initiated by user at *Time*. Version of requested configuration is *Num*.
From RECONFIGURING to RUNNING	INFO	The reconfiguration operation for instance group *InstanceGroupID* in the Amazon EMR cluster *ClusterId (ClusterName)* is complete. The reconfiguration started at *Time* and took *Num* minutes to complete. Current configuration version is *Num*.
From RUNNING to RECONFIGURING	INFO	A reconfiguration for instance group *InstanceGroupID* in the Amazon EMR cluster *ClusterId (ClusterName)* started at *Time*. It is configuring from version number *Num* to version number *Num*.
RESIZING	INFO	Reconfiguring operation towards configuration version *Num* for instance group *InstanceGroupID* in the Amazon EMR cluster *ClusterId (ClusterName)* is temporarily blocked at *Time* because instance group is in *State*.
RECONFIGURING	INFO	Resizing operation towards instance count *Num* for instance group *InstanceGroupID* in the Amazon EMR cluster *ClusterId (ClusterName)* is temporarily blocked at *Time* because the instance group is in *State*.
RECONFIGURING	WARNING	The reconfiguration operation for instance group *InstanceGroupID* in the Amazon EMR cluster *ClusterId*

State or state change	Severity	Message
		(ClusterName) failed at *Time* and took *Num* minutes to fail. Failed configuration version is *Num*.
RECONFIGURING	INFO	Configurations are reverting to the previous successful version number *Num*for instance group *InstanceGroupID* in the Amazon EMR cluster *ClusterId (ClusterName)* at *Time*. New configuration version is *Num*.
From RECONFIGURING to RUNNING	INFO	Configurations were successfully reverted to the previous successful version *Num* for instance group *InstanceGroupID* in the Amazon EMR cluster *ClusterId (ClusterName)* at *Time*. New configuration version is *Num*.
From RECONFIGURING to SUSPENDED	CRITICAL	Failed to revert to the previous successful version *Num* for Instance group *InstanceGroupID* in the Amazon EMR cluster *ClusterId (ClusterName)* at *Time*.

Automatic scaling policy events

State or state change	Severity	Message
PENDING	INFO	An Auto Scaling policy was added to instance group *InstanceGroupID* in Amazon EMR cluster *ClusterId (ClusterName)* at *Time*. The policy is pending attachment. - or - The Auto Scaling policy for instance group *InstanceGroupID* in Amazon EMR cluster *ClusterId (ClusterName)* was updated at *Time*. The policy is pending attachment.
ATTACHED	INFO	The Auto Scaling policy for instance group *InstanceGroupID* in Amazon EMR cluster *ClusterId*

State or state change	Severity	Message
		(ClusterName) was attached at *Time*.
DETACHED	INFO	The Auto Scaling policy for instance group *InstanceGroupID* in Amazon EMR cluster *ClusterId (ClusterName)* was detached at *Time*.
FAILED	ERROR	The Auto Scaling policy for instance group *InstanceGroupID* in Amazon EMR cluster *ClusterId (ClusterName)* could not attach and failed at *Time*. - or - The Auto Scaling policy for instance group *InstanceGroupID* in Amazon EMR cluster *ClusterId (ClusterName)* could not detach and failed at *Time*.

Step events

State or state change	Severity	Message
PENDING	INFO	Step *StepID (StepName)* was added to Amazon EMR cluster *ClusterId (ClusterName)* at *Time* and is pending execution.
CANCEL_PENDING	WARN	Step *StepID (StepName)* in Amazon EMR cluster *ClusterId (ClusterName)* was cancelled at *Time* and is pending cancellation.
RUNNING	INFO	Step *StepID (StepName)* in Amazon EMR cluster *ClusterId (ClusterName)* started running at *Time*.
COMPLETED	INFO	Step *StepID (StepName)* in Amazon EMR cluster *ClusterId (ClusterName)* completed execution at *Time*. The step started running at *Time* and took *Num* minutes to complete.

State or state change	Severity	Message
CANCELLED	WARN	Cancellation request has succeeded for cluster step *StepID (StepName)* in Amazon EMR cluster *ClusterId (ClusterName)* at *Time*, and the step is now cancelled.
FAILED	ERROR	Step *StepID (StepName)* in Amazon EMR cluster *ClusterId (ClusterName)* failed at *Time*.

View 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.

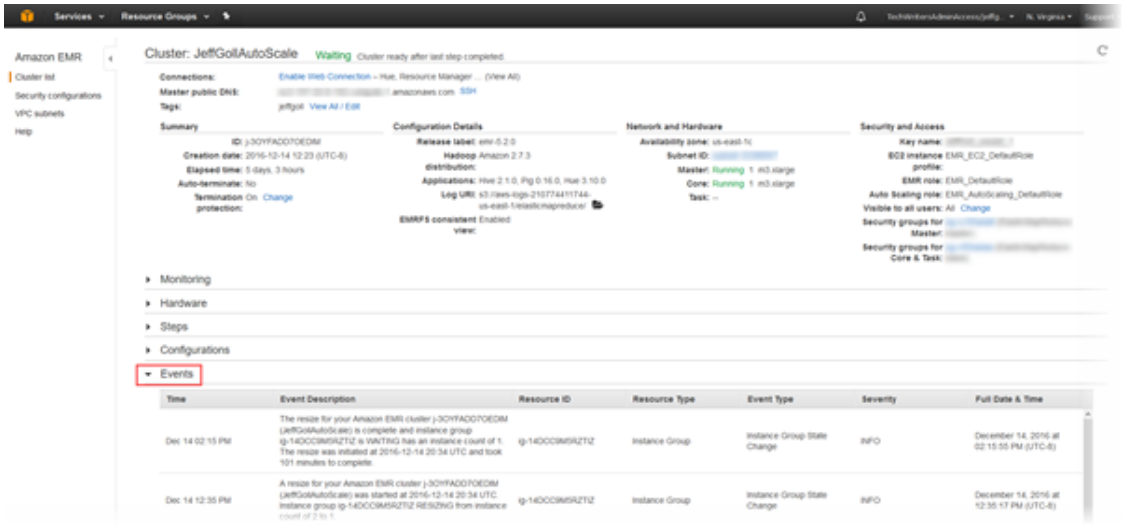
> **Note**
> If you don't want a user to see all cluster events for a region, add a statement that denies permission (`"Effect": "Deny"`) for the `elasticmapreduce:ViewEventsFromAllClustersInConsole` action to a policy that is attached to the user.

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.



Create 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*.

For information about setting up CloudWatch event rules, see Creating a CloudWatch rule that triggers on an event.

Monitor metrics with CloudWatch

Metrics are updated every five minutes and automatically collected and pushed to CloudWatch for every Amazon EMR cluster. This interval is not configurable. There is no charge for the Amazon EMR metrics reported in CloudWatch. These five minute datapoint metrics are archived for 63 days, after which the data is discarded.

How do I use Amazon EMR metrics?

The following table shows common uses for metrics reported by Amazon EMR. These are suggestions to get you started, not a comprehensive list. For a complete list of metrics reported by Amazon EMR, see Metrics reported by Amazon EMR in CloudWatch (p. 578).

How do I?	Relevant metrics
Track the progress of my cluster	Look at the `RunningMapTasks`, `RemainingMapTasks`, `RunningReduceTasks`, and `RemainingReduceTasks` metrics.
Detect clusters that are idle	The `IsIdle` 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 `MRUnhealthyNodes` metric tracks when one or more core or task nodes run out of local disk storage and transition to an `UNHEALTHY` YARN state. For example, core or task nodes are running low on disk space and will not be able to run tasks.
Detect when a cluster runs out of storage	The `HDFSUtilization` metric monitors the cluster's combined HDFS capacity, and can require resizing the cluster to add more core nodes. For example, the HDFS utilization is high, which may affect jobs and cluster health.
Detect when a cluster is running at reduced capacity	The `MRLostNodes` metric tracks when one or more core or task nodes is unable to communicate with the master node. For example, the core or task node is unreachable by the master node.

For more information, see Cluster terminates with NO_SLAVE_LEFT and core nodes FAILED_BY_MASTER (p. 666) and AWSSupport-AnalyzeEMRLogs.

Access CloudWatch metrics for Amazon EMR

You can view the metrics that Amazon EMR reports to CloudWatch using the Amazon EMR console or the CloudWatch console. You can also retrieve metrics using the CloudWatch CLI command `mon-get-stats` or the CloudWatch `GetMetricStatistics` API. For more information about viewing or retrieving metrics for Amazon EMR using CloudWatch, see the Amazon CloudWatch User Guide.

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**, or **IO** 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.

Set alarms on metrics

Amazon EMR pushes metrics to CloudWatch, which means you can use CloudWatch to set alarms on your Amazon EMR metrics. For example, you can configure an alarm in CloudWatch to send you an email any time the HDFS utilization rises above 80%. For detailed instructions, see Create or edit a CloudWatch alarm in the *Amazon CloudWatch User Guide*.

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
Cluster Status	
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 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. Use case: Monitor cluster performance

Metric	Description
	Units: *Boolean*
ContainerAllocated	The number of resource containers allocated by the ResourceManager. Use case: Monitor cluster progress Units: *Count*
ContainerReserved	The number of containers reserved. Use case: Monitor cluster progress Units: *Count*
ContainerPending	The number of containers in the queue that have not yet been allocated. Use case: Monitor cluster progress Units: *Count*
ContainerPendingRatio	The ratio of pending containers to containers allocated (ContainerPendingRatio = ContainerPending / ContainerAllocated). If ContainerAllocated = 0, then ContainerPendingRatio = ContainerPending. The value of ContainerPendingRatio represents a number, not a percentage. This value is useful for scaling cluster resources based on container allocation behavior. Units: *Count*
AppsCompleted	The number of applications submitted to YARN that have completed. Use case: Monitor cluster progress Units: *Count*
AppsFailed	The number of applications submitted to YARN that have failed to complete. Use case: Monitor cluster progress, Monitor cluster health Units: *Count*
AppsKilled	The number of applications submitted to YARN that have been killed. Use case: Monitor cluster progress, Monitor cluster health Units: *Count*
AppsPending	The number of applications submitted to YARN that are in a pending state. Use case: Monitor cluster progress Units: *Count*

Metric	Description
AppsRunning	The number of applications submitted to YARN that are running. Use case: Monitor cluster progress Units: *Count*
AppsSubmitted	The number of applications submitted to YARN. Use case: Monitor cluster progress Units: *Count*
Node Status	
CoreNodesRunning	The number of core nodes working. Data points for this metric are reported only when a corresponding instance group exists. Use case: Monitor cluster health Units: *Count*
CoreNodesPending	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. Use case: Monitor cluster health Units: *Count*
LiveDataNodes	The percentage of data nodes that are receiving work from Hadoop. Use case: Monitor cluster health Units: *Percent*
MRTotalNodes	The number of nodes presently available to MapReduce jobs. Equivalent to YARN metric `mapred.resourcemanager.TotalNodes`. Use ase: Monitor cluster progress Units: *Count*
MRActiveNodes	The number of nodes presently running MapReduce tasks or jobs. Equivalent to YARN metric `mapred.resourcemanager.NoOfActiveNodes`. Use case: Monitor cluster progress Units: *Count*

Metric	Description
MRLostNodes	The number of nodes allocated to MapReduce that have been marked in a LOST state. Equivalent to YARN metric `mapred.resourcemanager.NoOfLostNodes`. Use case: Monitor cluster health, Monitor cluster progress Units: *Count*
MRUnhealthyNodes	The number of nodes available to MapReduce jobs marked in an UNHEALTHY state. Equivalent to YARN metric `mapred.resourcemanager.NoOfUnhealthyNodes`. Use case: Monitor cluster progress Units: *Count*
MRDecommissionedNodes	The number of nodes allocated to MapReduce applications that have been marked in a DECOMMISSIONED state. Equivalent to YARN metric `mapred.resourcemanager.NoOfDecommissionedNodes`. Use ase: Monitor cluster health, Monitor cluster progress Units: *Count*
MRRebootedNodes	The number of nodes available to MapReduce that have been rebooted and marked in a REBOOTED state. Equivalent to YARN metric `mapred.resourcemanager.NoOfRebootedNodes`. Use case: Monitor cluster health, Monitor cluster progress Units: *Count*
MultiMasterInstanceGroupNodesRunning	The number of running master nodes. Use case: Monitor master node failure and replacement Units: *Count*
MultiMasterInstanceGroupNodesRunningPercentage	The percentage of master nodes that are running over the requested master node instance count. Use case: Monitor master node failure and replacement Units: *Percent*
MultiMasterInstanceGroupNodesRequested	The number of requested master nodes. Use case: Monitor master node failure and replacement Units: *Count*
IO	

Metric	Description
S3BytesWritten	The number of bytes written to Amazon S3. This metric aggregates MapReduce jobs only, and does not apply for other workloads on Amazon EMR. Use case: Analyze cluster performance, Monitor cluster progress Units: *Count*
S3BytesRead	The number of bytes read from Amazon S3. This metric aggregates MapReduce jobs only, and does not apply for other workloads on Amazon EMR. Use case: Analyze cluster performance, Monitor cluster progress Units: *Count*
HDFSUtilization	The percentage of HDFS storage currently used. Use case: Analyze cluster performance Units: *Percent*
HDFSBytesRead	The number of bytes read from HDFS. This metric aggregates MapReduce jobs only, and does not apply for other workloads on EMR. Use case: Analyze cluster performance, Monitor cluster progress Units: *Count*
HDFSBytesWritten	The number of bytes written to HDFS. This metric aggregates MapReduce jobs only, and does not apply for other workloads on EMR. Use case: Analyze cluster performance, Monitor cluster progress Units: *Count*
MissingBlocks	The number of blocks in which HDFS has no replicas. These might be corrupt blocks. Use case: Monitor cluster health Units: *Count*
CorruptBlocks	The number of blocks that HDFS reports as corrupted. Use case: Monitor cluster health Units: *Count*

Metric	Description
TotalLoad	The total number of concurrent data transfers. Use case: Monitor cluster health Units: *Count*
MemoryTotalMB	The total amount of memory in the cluster. Use case: Monitor cluster progress Units: *Count*
MemoryReservedMB	The amount of memory reserved. Use case: Monitor cluster progress Units: *Count*
MemoryAvailableMB	The amount of memory available to be allocated. Use case: Monitor cluster progress Units: *Count*
YARNMemoryAvailablePercentage	The percentage of remaining memory available to YARN (YARNMemoryAvailablePercentage = MemoryAvailableMB / MemoryTotalMB). This value is useful for scaling cluster resources based on YARN memory usage. Units: *Percent*
MemoryAllocatedMB	The amount of memory allocated to the cluster. Use case: Monitor cluster progress Units: *Count*
PendingDeletionBlocks	The number of blocks marked for deletion. Use case: Monitor cluster progress, Monitor cluster health Units: *Count*
UnderReplicatedBlocks	The number of blocks that need to be replicated one or more times. Use case: Monitor cluster progress, Monitor cluster health Units: *Count*
DfsPendingReplicationBlocks	The status of block replication: blocks being replicated, age of replication requests, and unsuccessful replication requests. Use case: Monitor cluster progress, Monitor cluster health Units: *Count*

Metric	Description
CapacityRemainingGB	The amount of remaining HDFS disk capacity. Use case: Monitor cluster progress, Monitor cluster health Units: *Count*

The following are Hadoop 1 metrics:

Metric	Description
Cluster Status	
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 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. Use case: Monitor cluster performance Units: *Boolean*
JobsRunning	The number of jobs in the cluster that are currently running. Use case: Monitor cluster health Units: *Count*
JobsFailed	The number of jobs in the cluster that have failed. Use case: Monitor cluster health Units: *Count*
Map/Reduce	
MapTasksRunning	The number of running map tasks for each job. If you have a scheduler installed and multiple jobs running, multiple graphs are generated. Use case: Monitor cluster progress Units: *Count*
MapTasksRemaining	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. Use case: Monitor cluster progress Units: *Count*

Metric	Description
MapSlotsOpen	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. Use case: Analyze cluster performance Units: *Count*
RemainingMapTasksPerSlot	The ratio of the total map tasks remaining to the total map slots available in the cluster. Use case: Analyze cluster performance Units: *Ratio*
ReduceTasksRunning	The number of running reduce tasks for each job. If you have a scheduler installed and multiple jobs running, multiple graphs are generated. Use case: Monitor cluster progress Units: *Count*
ReduceTasksRemaining	The number of remaining reduce tasks for each job. If you have a scheduler installed and multiple jobs running, multiple graphs are generated. Use case: Monitor cluster progress Units: *Count*
ReduceSlotsOpen	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. Use case: Analyze cluster performance Units: *Count*
Node Status	
CoreNodesRunning	The number of core nodes working. Data points for this metric are reported only when a corresponding instance group exists. Use case: Monitor cluster health Units: *Count*
CoreNodesPending	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. Use case: Monitor cluster health Units: *Count*

Metric	Description
LiveDataNodes	The percentage of data nodes that are receiving work from Hadoop. Use case: Monitor cluster health Units: *Percent*
TaskNodesRunning	The number of task nodes working. Data points for this metric are reported only when a corresponding instance group exists. Use case: Monitor cluster health Units: *Count*
TaskNodesPending	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. Use case: Monitor cluster health Units: *Count*
LiveTaskTrackers	The percentage of task trackers that are functional. Use case: Monitor cluster health Units: *Percent*
IO	
S3BytesWritten	The number of bytes written to Amazon S3. This metric aggregates MapReduce jobs only, and does not apply for other workloads on Amazon EMR. Use case: Analyze cluster performance, Monitor cluster progress Units: *Count*
S3BytesRead	The number of bytes read from Amazon S3. This metric aggregates MapReduce jobs only, and does not apply for other workloads on Amazon EMR. Use case: Analyze cluster performance, Monitor cluster progress Units: *Count*
HDFSUtilization	The percentage of HDFS storage currently used. Use case: Analyze cluster performance Units: *Percent*

Metric	Description
HDFSBytesRead	The number of bytes read from HDFS. Use case: Analyze cluster performance, Monitor cluster progress Units: *Count*
HDFSBytesWritten	The number of bytes written to HDFS. Use case: Analyze cluster performance, Monitor cluster progress Units: *Count*
MissingBlocks	The number of blocks in which HDFS has no replicas. These might be corrupt blocks. Use case: Monitor cluster health Units: *Count*
TotalLoad	The current, total number of readers and writers reported by all DataNodes in a cluster. 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. Units: *Count*

Cluster capacity metrics

The following metrics indicate the current or target capacities of a cluster. These metrics are only available when managed scaling or auto-termination is enabled.

For clusters composed of instance fleets, the cluster capacity metrics are measured in `Units`. For clusters composed of instance groups, the cluster capacity metrics are measured in `Nodes` or `VCPU` based on the unit type used in the managed scaling policy. For more information, see Using EMR-managed scaling in the *Amazon EMR Management Guide*.

Metric	Description
• `TotalUnitsRequested` • `TotalNodesRequested` • `TotalVCPURequested`	The target total number of units/nodes/vCPUs in a cluster as determined by managed scaling. Units: *Count*
• `TotalUnitsRunning` • `TotalNodesRunning` • `TotalVCPURunning`	The current total number of units/nodes/vCPUs available in a running cluster. When a cluster resize is requested, this metric will be updated after the new instances are added or removed from the cluster.

Metric	Description
	Units: *Count*
• `CoreUnitsRequested` • `CoreNodesRequested` • `CoreVCPURequested`	The target number of CORE units/nodes/vCPUs in a cluster as determined by managed scaling. Units: *Count*
• `CoreUnitsRunning` • `CoreNodesRunning` • `CoreVCPURunning`	The current number of CORE units/nodes/vCPUs running in a cluster. Units: *Count*
• `TaskUnitsRequested` • `TaskNodesRequested` • `TaskVCPURequested`	The target number of TASK units/nodes/vCPUs in a cluster as determined by managed scaling. Units: *Count*
• `TaskUnitsRunning` • `TaskNodesRunning` • `TaskVCPURunning`	The current number of TASK units/nodes/vCPUs running in a cluster. Units: *Count*

Amazon EMR emits the following metrics at a one-minute granularity when you enable auto-termination using an auto-termination policy. Some metrics are only available for Amazon EMR versions 6.4.0 and later. To learn more about auto-termination, see Using an auto-termination policy (p. 169).

Metric	Description
`TotalNotebookKernels`	The total number of running and idle notebook kernels on the cluster. This metric is only available for Amazon EMR versions 6.4.0 and later.
`AutoTerminationIsClusterIdle`	Indicates whether the cluster is in use. A value of **0** indicates that the cluster is in active use by one of the following components: • A YARN application • HDFS • A notebook • An on-cluster UI, such as the Spark History Server A value of **1** indicates that the cluster is idle. Amazon EMR checks for continuous cluster idleness (`AutoTerminationIsClusterIdle` = 1). When a cluster's idle time equals the `IdleTimeout` value in your auto-termination policy, Amazon EMR terminates the cluster.

Dimensions for Amazon EMR metrics

Amazon EMR data can be filtered using any of the dimensions in the following table.

Dimension	Description
JobFlowId	The same as cluster ID, which is the unique identifier of a cluster in the form `j-XXXXXXXXXXXXX`. Find this value by clicking on the cluster in the Amazon EMR console.

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.

In the case where a process, rather than an IAM user, creates a cluster, you can use the `principalId` identifier to determine the user associated with the cluster's creation. 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",
            "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":"m5.xlarge",
               "ec2KeyName":"emr-integtest",
               "instanceCount":1,
               "masterInstanceType":"m5.xlarge",
               "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. 448). 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

- Before you connect: Authorize inbound traffic (p. 592)
- Connect to the master node using SSH (p. 593)

Before you connect: Authorize inbound traffic

Before you connect to an Amazon EMR cluster, you must authorize inbound SSH traffic (port 22) from trusted clients such as your computer's IP address. In order to do so, edit the managed security group rules for the nodes to which you want to connect. For example, the following instructions show you how to add an inbound rule for SSH access to the default ElasticMapReduce-master security group.

For more information about using security groups with Amazon EMR, see Control network traffic with security groups (p. 532).

To allow SSH access for trusted sources for the ElasticMapReduce-master security group with the old console

To edit your security groups, you must have permission to manage security groups for the VPC that the cluster is in. For more information, see Changing Permissions for an IAM User and the Example Policy that allows managing EC2 security groups in the *IAM User Guide*.

1. Open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Choose **Clusters**. Choose the **Name** of the cluster you want to modify.
3. Choose the **Security groups for Master** link under **Security and access**.
4. Choose **ElasticMapReduce-master** from the list.
5. Choose the **Inbound rules** tab and then **Edit inbound rules**.
6. Check for an inbound rule that allows public access with the following settings. If it exists, choose **Delete** to remove it.

   - **Type**

     SSH
   - **Port**

     22
   - **Source**

     Custom 0.0.0.0/0

     Warning
     Before December 2020, the ElasticMapReduce-master security group had a pre-configured rule to allow inbound traffic on Port 22 from all sources. This rule was created to simplify initial SSH connections to the master node. We strongly recommend that you remove this inbound rule and restrict traffic to trusted sources.
7. Scroll to the bottom of the list of rules and choose **Add Rule**.
8. For **Type**, select **SSH**.

   Selecting SSH automatically enters **TCP** for **Protocol** and **22** for **Port Range**.
9. For source, select **My IP** to automatically add your IP address as the source address. You can also add a range of **Custom** trusted client IP addresses, or create additional rules for other clients. Many network environments dynamically allocate IP addresses, so you might need to update your IP addresses for trusted clients in the future.
10. Choose **Save**.
11. Optionally, choose **ElasticMapReduce-slave** from the list and repeat the steps above to allow SSH client access to core and task nodes.

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. 599).

To connect to the master node using SSH, you need the public DNS name of the master node. In addition, the security group associated with the master node must have an inbound rule that allows SSH (TCP port 22) traffic from a source that includes the client where the SSH connection originates. You may need to add a rule to allow an SSH connection from your client. For more information about modifying security group rules, see Control network traffic with security groups (p. 532) and Adding rules to a security group in the *Amazon EC2 User Guide for Linux Instances*.

Retrieve the public DNS name of the master node

You can retrieve the master public DNS name using the Amazon EMR console and the AWS CLI.

Console

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/emr.
2. On the **Cluster List** page, select the link for your cluster.
3. Note the **Master public DNS** value that appears in the **Summary** section 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.

CLI

To retrieve the public DNS name of the master node using the AWS CLI

1. To retrieve the cluster identifier, type the following command.

   ```
   aws emr list-clusters
   ```

   The output lists your clusters including the cluster IDs. Note the cluster ID for the cluster to which you are connecting.

   ```
   "Status": {
       "Timeline": {
           "ReadyDateTime": 1408040782.374,
   ```

```
        "CreationDateTime": 1408040501.213
    },
    "State": "WAITING",
    "StateChangeReason": {
        "Message": "Waiting after step completed"
    }
},
"NormalizedInstanceHours": 4,
"Id": "j-2AL4XXXXXX5T9",
"Name": "My cluster"
```

2. To list the cluster instances including the master public DNS name for the cluster, type one of the following commands. Replace *j-2AL4XXXXXX5T9* with the cluster ID returned by the previous command.

```
aws emr list-instances --cluster-id j-2AL4XXXXXX5T9
```

Or:

```
aws emr describe-cluster --cluster-id j-2AL4XXXXXX5T9
```

The output lists the cluster instances including DNS names and IP addresses. Note the value for `PublicDnsName`.

```
"Status": {
    "Timeline": {
        "ReadyDateTime": 1408040779.263,
        "CreationDateTime": 1408040515.535
    },
    "State": "RUNNING",
    "StateChangeReason": {}
},
"Ec2InstanceId": "i-e89b45e7",
"PublicDnsName": "ec2-###-##-##-###.us-west-2.compute.amazonaws.com"

"PrivateDnsName": "ip-###-##-##-###.us-west-2.compute.internal",
"PublicIpAddress": "##.###.###.##",
"Id": "ci-12XXXXXXXXFMH",
"PrivateIpAddress": "###.##.#.###"
```

For more information, see Amazon EMR commands in the AWS CLI.

Connect to the master node using SSH 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. Ensure you've allowed inbound SSH traffic. For instructions, see Before you connect: Authorize inbound traffic (p. 592).
2. 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.
3. Type the following command to set the permissions. Replace *~/mykeypair.pem* with the full path and file name of your key pair private key file. For example `C:\Users\<username>\.ssh\mykeypair.pem`.

   ```
   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 full path and file name of your `.pem` file. For example `C:\Users\<username>\.ssh\mykeypair.pem`.

   ```
   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
   ```

If you're experiencing difficulty with using SSH to connect to your master node, see Troubleshoot connecting to your instance.

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. Ensure you've allowed inbound SSH traffic. For instructions, see Before you connect: Authorize inbound traffic (p. 592).
2. Open `putty.exe`. You can also launch PuTTY from the Windows programs list.
3. If necessary, in the **Category** list, choose **Session**.
4. For **Host Name (or IP address)**, type hadoop@*`MasterPublicDNS`*. For example: hadoop@*`ec2-###-##-##-###.compute-1.amazonaws.com`*.
5. In the **Category** list, choose **Connection > SSH**, **Auth**.
6. For **Private key file for authentication**, choose **Browse** and select the `.ppk` file that you generated.
7. 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 .
8. 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**.

If you're experiencing difficulty with using SSH to connect to your master node, see Troubleshoot connecting to your instance.

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. 595).

To connect to the master node using the AWS CLI

1. Ensure you've allowed inbound SSH traffic. For instructions, see Before you connect: Authorize inbound traffic (p. 592).
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 connection to the master node. In the following example, replace *j-2AL4XXXXXX5T9* with the cluster ID and replace *~/mykeypair.key* with the full path and file name of your `.pem` file (for Linux, Unix, and Mac OS X) or `.ppk` file (for Windows). For example `C:\Users\<username>\.ssh\mykeypair.pem`.

```
aws emr ssh --cluster-id j-2AL4XXXXXX5T9 --key-pair-file ~/mykeypair.key
```

4. 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. If you're experiencing difficulty with using SSH to connect to your master node, see Troubleshoot connecting to your instance.

Amazon EMR service ports

Note
The following are interfaces and service ports for components on Amazon EMR. This is not a complete list of service ports. Non-default services, such as SSL ports and different types of protocols, are not listed.

Important
Use caution when you edit security group rules to open ports. Be sure to add rules that only allow traffic from trusted and authenticated clients for the protocols and ports that are required to run your workloads.

Component	Service description	Service running by default	Port	Configuration key
Hadoop	HTTP KMS REST API	Yes	9600	hadoop.kms.http.port
HDFS	Namenode Web UI	Yes	9870	dfs.namenode.http-address
	Namenode RPC	Yes	8020	dfs.namenode.rpc-address
	DataNode Web UI	Yes	9864	dfs.datanode.http.address
	Datanode HTTP for data transfer	Yes	9866	dfs.datanode.address
	Datanode RPC for data transfer	Yes	9867	dfs.datanode.ipc.address
Hive	HiveServer2 Thrift	Yes	10000	hive.server2.thrift.port
	HiveServer2 HTTP	No	10001	hive.server2.thrift.http.port
	HiveServer2 Web UI	Yes	10002	hive.server2.webui.port

Component	Service description	Service running by default	Port	Configuration key
	Hive Metastore	Yes	9083	hive.metastore.port / metastore.thrift.port
	WebHCat	No	50111	templeton.port
	LLAP daemon management service (RPC)	No	15004	hive.llap.management.rpc.port
	YARN shuffle port for LLAP-daemon-hosted shuffle	No	15551	hive.llap.daemon.yarn.shuffle.port
	The LLAP daemon RPC	No	Dynamic	hive.llap.daemon.rpc.port
	LLAP daemon Web UI	No	15002	hive.llap.daemon.web.port
	LLAP daemon output service	No	15003	hive.llap.daemon.output.service.po
Oozie		Yes	11000	
Tez	Tez UI	Yes	8080	
YARN	Shuffle	Yes	13562	mapreduce.shuffle.port
	Localizer RPC	Yes	8040	yarn.nodemanager.localizer.addres
		Yes	8041	
	NM Webapp address	Yes	8042	yarn.nodemanager.webapp.addres
	RM web application	Yes	8088	yarn.resourcemanager.webapp.add
		Yes	8025	
	Scheduler	Yes	8030	yarn.resourcemanager.scheduler.a
	applications manager interface	Yes	8032	yarn.resourcemanager.address
	RM admin interface	Yes	8033	yarn.resourcemanager.admin.addr
	JobHistory Server Web UI	Yes	19888	mapreduce.jobhistory.webapp.add
	JobHistory Server Admin Web UI	Yes	10033	mapreduce.jobhistory.admin.addre
	JobHistory Server (RPC)	Yes	10020	mapreduce.jobhistory.address
	Application Timeline Server (RPC)	Yes	10200	yarn.timeline-service.address

Component	Service description	Service running by default	Port	Configuration key
	Application Timeline Server HTTP Web UI	Yes	8188	yarn.timeline-service.webapp.address
	Application Timeline Server HTTPS Web UI	No	8190	yarn.timeline-service.webapp.https.address
		Yes	20888	
Zookeeper	Client port	Yes	2181	
		Yes	37301	
		Yes	8341	

View web interfaces hosted on Amazon EMR clusters

Important
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. 532).

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. 593). 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.

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.

Name of interface	URI
Flink history server (EMR version 5.33 and later)	http://*master-public-dns-name*:8082/
Ganglia	http://*master-public-dns-name*/ganglia/
Hadoop HDFS NameNode (EMR version pre-6.x)	https://*master-public-dns-name*:50470/
Hadoop HDFS NameNode	http://*master-public-dns-name*:50070/
Hadoop HDFS DataNode	http://*coretask-public-dns-name*:50075/
Hadoop HDFS NameNode (EMR version 6.x)	https://*master-public-dns-name*:9870/

Name of interface	URI
Hadoop HDFS DataNode (EMR version pre-6.x)	https://*coretask-public-dns-name*:50475/
Hadoop HDFS DataNode (EMR version 6.x)	https://*coretask-public-dns-name*:9865/
HBase	http://*master-public-dns-name*:16010/
Hue	http://*master-public-dns-name*:8888/
JupyterHub	https://*master-public-dns-name*:9443/
Livy	http://*master-public-dns-name*:8998/
Spark HistoryServer	http://*master-public-dns-name*:18080/
Tez	http://*master-public-dns-name*:8080/tez-ui
YARN NodeManager	http://*coretask-public-dns-name*:8042/
YARN ResourceManager	http://*master-public-dns-name*:8088/
Zeppelin	http://*master-public-dns-name*:8890/

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 for Firefox or SwitchyOmega for Chrome to manage your SOCKS proxy settings. This method lets you automatically filter URLs based on text patterns and limit the proxy settings to domains that match the form of the master node's DNS name. 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. 604).

Note
If you modify the port where an application runs via cluster configuration, the hyperlink to the port will not update in the Amazon EMR console. This is because the console doesn't have the functionality to read `server.port` configuration.

With Amazon EMR version 5.25.0 or later, you can access Spark history server UI from the console without setting up a web proxy through an SSH connection. For more information, see One-click access to persistent Spark history server.

Topics

- Option 1: Set up an SSH tunnel to the master node using local port forwarding (p. 601)
- Option 2, part 1: Set up an SSH tunnel to the master node using dynamic port forwarding (p. 602)
- Option 2, part 2: Configure proxy settings to view websites hosted on the master node (p. 604)

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. 593). 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. 599).

Set up an SSH tunnel to the master node using local port forwarding with OpenSSH

To set up an SSH tunnel using local port forwarding in terminal

1. Ensure you've allowed inbound SSH traffic. For instructions, see Before you connect: Authorize inbound traffic (p. 592).
2. Open a terminal window. On Mac OS X, choose **Applications > Utilities > Terminal**. On other Linux distributions, terminal is typically found at **Applications > Accessories > Terminal**.
3. Type the following command to open an SSH tunnel on your local machine. This example command accesses the ResourceManager web interface by forwarding traffic on local port 8157 (a randomly chose 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. To access a different web interface, replace `8088` with the appropriate port number. For example, replace `8088` with `8890` for the Zeppelin interface.

   ```
   ssh -i ~/mykeypair.pem -N -L 8157:ec2-###-##-##-
   ###.compute-1.amazonaws.com:8088 hadoop@ec2-###-##-##-###.compute-1.amazonaws.com
   ```

   `-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.

   After you issue this command, the terminal remains open and does not return a response.

4. To open the ResourceManager web interface in your browser, type `http://localhost:8157/` in the address bar.
5. 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 SwitchyOmega 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. 593). 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. 599).

Set up an SSH tunnel to the master node using dynamic port forwarding with OpenSSH

To set up an SSH tunnel using dynamic port forwarding with OpenSSH

1. Ensure you've allowed inbound SSH traffic. For instructions, see Before you connect: Authorize inbound traffic (p. 592).
2. Open a terminal window. On Mac OS X, choose **Applications > Utilities > Terminal**. On other Linux distributions, terminal is typically found at **Applications > Accessories > Terminal**.
3. Type the following command to open an SSH tunnel on your local machine. Replace *~/mykeypair.pem* with the location and file name of your `.pem` file, replace *8157* with an unused, local port number, and replace *c2-###-##-##-###.compute-1.amazonaws.com* with the master public DNS name of your cluster.

   ```
   ssh -i ~/mykeypair.pem -N -D 8157 hadoop@ec2-###-##-##-###.compute-1.amazonaws.com
   ```

   After you issue this command, the terminal remains open and does not return a response.

   > **Note**
   > `-D` signifies the use of dynamic port forwarding which allows you to specify a local port used to forward data to all remote ports on the master node's local web server. Dynamic port forwarding creates a local SOCKS proxy listening on the port specified in the command.

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. 604).
5. 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. 595). 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. Ensure you've allowed inbound SSH traffic. For instructions, see Before you connect: Authorize inbound traffic (p. 592).
2. Create an SSH connection with the master node as shown in Connect to the master node using the AWS CLI (p. 596).
3. 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"
   ```

4. 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.
5. 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. 604).
6. 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 PuTTY

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 using PuTTY

1. Ensure you've allowed inbound SSH traffic. For instructions, see Before you connect: Authorize inbound traffic (p. 592).
2. 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**.
3. If necessary, in the **Category** list, choose **Session**.
4. In the **Host Name** field, type **hadoop@***`MasterPublicDNS`*. For example: **hadoop@***`ec2-###-##-##-###.compute-1.amazonaws.com`*.
5. In the **Category** list, expand **Connection > SSH**, and then choose **Auth**.
6. 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.
7. In the **Category** list, expand **Connection > SSH**, and then choose **Tunnels**.
8. In the **Source port** field, type `8157` (an unused local port), and then choose **Add**.
9. Leave the **Destination** field blank.
10. Select the **Dynamic** and **Auto** options.
11. Choose **Open**.
12. 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.
13. 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. 604).
14. 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 SwitchyOmega.

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. 602). For more information about the available web interfaces, see View web interfaces hosted on Amazon EMR clusters (p. 599).

Include the following settings when you set up your proxy add-on:

- Use **localhost** as the host address.
- Use the same local port number that you selected 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. 602). For example, port *8157*. This port must also match the port number you use in PuTTY or any other terminal emulator you use to connect.
- Specify the **SOCKS v5** protocol. SOCKS v5 lets you optionally set up user authorization.
- **URL Patterns**

  The following URL patterns should be allow-listed and specified with a wildcard pattern type:
  - The ***ec2*.amazonaws.com*** and ***10*.amazonaws.com*** patterns to match the public DNS name of clusters in US regions.
  - The ***ec2*.compute*** and ***10*.compute*** patterns to match the public DNS name of clusters in all other regions.
  - A **10.*** pattern to provide access to the JobTracker log files in Hadoop. Alter this filter if it conflicts with your network access plan.
  - The ***.ec2.internal*** and ***.compute.internal*** patterns to match the private (internal) DNS names of clusters in the `us-east-1` region and all other regions, respectively.

Example: Configure FoxyProxy for Firefox

The following example demonstrates a FoxyProxy Standard (version 7.5.1) configuration for Mozilla Firefox.

FoxyProxy provides a set of proxy management tools. It lets you use a proxy server for URLs that match patterns corresponding to domains used by the Amazon EC2 instances in your Amazon EMR cluster.

To install and configure FoxyProxy using Mozilla Firefox

1. In Firefox, go to https://addons.mozilla.org/, search for FoxyProxy Standard, and follow the instructions to add FoxyProxy to Firefox.
2. Using a text editor, create a JSON file named `foxyproxy-settings.json` from the following example configuration.

```
{
  "k20d21508277536715": {
    "active": true,
    "address": "localhost",
    "port": 8157,
    "username": "",
    "password": "",
    "type": 3,
    "proxyDNS": true,
    "title": "emr-socks-proxy",
    "color": "#0055E5",
    "index": 9007199254740991,
    "whitePatterns": [
      {
        "title": "*ec2*.amazonaws.com*",
        "active": true,
        "pattern": "*ec2*.amazonaws.com*",
        "importedPattern": "*ec2*.amazonaws.com*",
        "type": 1,
        "protocols": 1
      },
      {
```

```
        "title": "*ec2*.compute*",
        "active": true,
        "pattern": "*ec2*.compute*",
        "importedPattern": "*ec2*.compute*",
        "type": 1,
        "protocols": 1
      },
      {
        "title": "10.*",
        "active": true,
        "pattern": "10.*",
        "importedPattern": "http://10.*",
        "type": 1,
        "protocols": 2
      },
      {
        "title": "*10*.amazonaws.com*",
        "active": true,
        "pattern": "*10*.amazonaws.com*",
        "importedPattern": "*10*.amazonaws.com*",
        "type": 1,
        "protocols": 1
      },
      {
        "title": "*10*.compute*",
        "active": true,
        "pattern": "*10*.compute*",
        "importedPattern": "*10*.compute*",
        "type": 1,
        "protocols": 1
      },
      {
        "title": "*.compute.internal*",
        "active": true,
        "pattern": "*.compute.internal*",
        "importedPattern": "*.compute.internal*",
        "type": 1,
        "protocols": 1
      },
      {
        "title": "*.ec2.internal* ",
        "active": true,
        "pattern": "*.ec2.internal*",
        "importedPattern": "*.ec2.internal*",
        "type": 1,
        "protocols": 1
      }
    ],
    "blackPatterns": []
  },
  "logging": {
    "size": 100,
    "active": false
  },
  "mode": "patterns",
  "browserVersion": "68.12.0",
  "foxyProxyVersion": "7.5.1",
  "foxyProxyEdition": "standard"
}
```

3. Open the Firefox **Manage Your Extensions** page (go to **about:addons**, then choose **Extensions**).
4. Choose **FoxyProxy Standard**, then choose the more options button (the button that looks like an ellipsis).
5. Select **Options** from the dropdown.

6. Choose **Import Settings** from the left menu.
7. On the **Import Settings** page, choose **Import Settings** under **Import Settings from FoxyProxy 6.0+**, browse to the location of the `foxyproxy-settings.json` file you created, select the file, and choose **Open**.
8. Choose **OK** when prompted to overwrite the existing settings and save your new configuration.

Example: Configure SwitchyOmega for chrome

The following example demonstrates how to set up the SwitchyOmega extension for Google Chrome. SwitchyOmega lets you configure, manage, and switch between multiple proxies.

To install and configure SwitchyOmega using Google Chrome

1. Go to https://chrome.google.com/webstore/category/extensions, search for **Proxy SwitchyOmega**, and add it to Chrome.
2. Choose **New profile** and enter `emr-socks-proxy` as the profile name.
3. Choose **PAC profile** and then **Create**. Proxy Auto-Configuration (PAC) files help you define an allow list for browser requests that should be forwarded to a web proxy server.
4. In the **PAC Script** field, replace the contents with the following script that defines which URLs should be forwarded through your web proxy server. If you specified a different port number when you set up your SSH tunnel, replace *8157* with your port number.

```
function FindProxyForURL(url, host) {
    if (shExpMatch(url, "*ec2*.amazonaws.com*")) return 'SOCKS5 localhost:8157';
    if (shExpMatch(url, "*ec2*.compute*")) return 'SOCKS5 localhost:8157';
    if (shExpMatch(url, "http://10.*")) return 'SOCKS5 localhost:8157';
    if (shExpMatch(url, "*10*.compute*")) return 'SOCKS5 localhost:8157';
    if (shExpMatch(url, "*10*.amazonaws.com*")) return 'SOCKS5 localhost:8157';
    if (shExpMatch(url, "*.compute.internal*")) return 'SOCKS5 localhost:8157';
    if (shExpMatch(url, "*ec2.internal*")) return 'SOCKS5 localhost:8157';
    return 'DIRECT';
}
```

5. Under **Actions**, choose **Apply changes** to save your proxy settings.
6. On the Chrome toolbar, choose SwitchyOmega and select the `emr-socks-proxy` profile.

Access a web interface in the browser

To open a web interface, enter the public DNS name of your master or core node followed by the port number for your chosen interface into your browser address bar. The following example shows the URL you would enter to connect to the Spark HistoryServer.

```
http://master-public-dns-name:18080/
```

For instructions on retrieving the public DNS name of a node, see Retrieve the public DNS name of the master node (p. 593). For a complete list of web interface URLs, see View web interfaces hosted on Amazon EMR clusters (p. 599).

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. 167). 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.

> **Note**
> You can't restart a terminated cluster, but you can clone a terminated cluster to reuse its configuration for a new cluster. For more information, see Cloning a cluster using the console (p. 641).

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-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.

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 Amazon EMR cluster automatically or manually in response to workloads that have varying demands. To use automatic scaling, you have two options. You can enable EMR managed scaling or create a custom automatic scaling policy. The following table describes the differences between the two options.

	EMR managed scaling	Custom automatic scaling
Scaling policies and rules	No policy required. EMR manages the automatic scaling activity by continuously evaluating cluster metrics and making optimized scaling decisions.	You need to define and manage the automatic scaling policies and rules, such as the specific conditions that trigger scaling activities, evaluation periods, cooldown periods, etc.
Supported EMR release versions	Amazon EMR version 5.30.0 and later (except Amazon EMR version 6.0.0)	Amazon EMR version 4.0.0 and later
Supported cluster composition	Instance groups or instance fleets	Instance groups only
Scaling limits configuration	Scaling limits are configured for the entire cluster.	Scaling limits can only be configured for each instance group.
Metrics evaluation frequency	Every 5 to 10 seconds More frequent evaluation of metrics allows EMR to make more precise scaling decisions.	You can define the evaluation periods only in five-minute increments.
Supported applications	Only YARN applications are supported, such as Spark, Hadoop, Hive, Flink. Other applications, such as Presto and HBase, are currently not supported.	You can choose which applications are supported when defining the automatic scaling rules.

Considerations

- An Amazon EMR cluster always consists of one or three master nodes. You can't scale the number of master nodes after you initially configure the cluster. You can only scale core and task nodes in a cluster.
- Reconfiguration and resizing of an instance group cannot occur at the same time. If a reconfiguration is initiated while an instance group is resizing, then reconfiguration cannot start until the instance group has completed resizing, and the other way around.

Using EMR managed scaling in Amazon EMR

Topics

- Understanding node allocation strategy and scenarios (p. 613)

- Understanding managed scaling metrics (p. 617)
- Use the AWS Management Console to configure managed scaling (p. 621)
- Using the AWS CLI to configure managed scaling (p. 622)
- Using AWS SDK for Java to configure managed scaling (p. 623)

> **Important**
> We strongly recommend that you use the latest Amazon EMR releases for managed scaling. In earlier Amazon EMR versions, you could experience intermittent application failures or delay in scaling. This issue was fixed in Amazon EMR 5.30.2, 5.31.1, 5.32.1, 5.33.1 and 6.0.1, 6.1.1, 6.2.1, 6.3.1.

With Amazon EMR versions 5.30.0 and later (except for Amazon EMR 6.0.0), you can enable EMR managed scaling. Managed scaling lets you automatically increase or decrease the number of instances or units in your cluster based on workload. EMR continuously evaluates cluster metrics to make scaling decisions that optimize your clusters for cost and speed. Managed scaling is available for clusters composed of either instance groups or instance fleets.

You need to configure the following parameters for managed scaling. The limit only applies to the core and task nodes. The master node cannot be scaled after initial configuration.

- **Minimum** (`MinimumCapacityUnits`) – The lower boundary of allowed EC2 capacity in a cluster. It is measured through virtual central processing unit (vCPU) cores or instances for instance groups. It is measured through units for instance fleets.
- **Maximum** (`MaximumCapacityUnits`) – The upper boundary of allowed EC2 capacity in a cluster. It is measured through virtual central processing unit (vCPU) cores or instances for instance groups. It is measured through units for instance fleets.
- **On-Demand limit** (`MaximumOnDemandCapacityUnits`) (Optional) – The upper boundary of allowed EC2 capacity for On-Demand market type in a cluster. If this parameter is not specified, it defaults to the value of `MaximumCapacityUnits`.

  This parameter is used to split capacity allocation between On-Demand and Spot Instances. For example, if you set the minimum parameter as 2 instances, the maximum parameter as 100 instances, the On-Demand limit as 10 instances, then EMR managed scaling scales up to 10 On-Demand Instances and allocates the remaining capacity to Spot Instances. For more information, see Node allocation scenarios (p. 614).
- **Maximum core nodes** (`MaximumCoreCapacityUnits`) (Optional) – The upper boundary of allowed EC2 capacity for core node type in a cluster. If this parameter is not specified, it defaults to the value of `MaximumCapacityUnits`.

  This parameter is used to split capacity allocation between core and task nodes. For example, if you set the minimum parameter as 2 instances, the maximum as 100 instances, the maximum core node as 17 instances, then EMR managed scaling scales up to 17 core nodes and allocates the remaining 83 instances to task nodes. For more information, see Node allocation scenarios (p. 614).

For more information about managed scaling parameters, see `ComputeLimits`.

Feature history

This section lists recent updates to the Amazon EMR managed scaling feature and the release version that contains the update.

Managed scaling feature history

Release date	Feature	Amazon EMR versions
March 21, 2022	Added Spark shuffle data awareness used when scaling-down clusters. For Amazon EMR clusters with Apache Spark and the managed scaling feature enabled, Amazon EMR continuously monitors Spark executors and intermediate shuffle data locations. Using this information, Amazon EMR scales-down only under-utilized instances which don't contain actively used shuffle data. This prevents recomputation of lost shuffle data, helping to lower cost and improve job performance. For more information, see the Spark Programming Guide.	5.34.0 and later, 6.4.0 and later

Considerations and limitations

- To use managed scaling, the metrics-collector process must be able to connect to the public API endpoint for managed scaling in API Gateway. If you use a private DNS name with Amazon Virtual Private Cloud, managed scaling won't function properly. To ensure that managed scaling works, we recommend that you take one of the following actions:
  - Remove the API Gateway interface VPC endpoint from your Amazon VPC.
  - Follow the instructions in Why do I get an HTTP 403 Forbidden error when connecting to my API Gateway APIs from a VPC? to disable the private DNS name setting.
  - Launch your cluster in a private subnet instead. For more information, see the topic on Private subnets (p. 308).
- If your YARN jobs are intermittently slow during scale down and YARN Resource Manager logs show that most of your nodes were deny listed during that time, you can adjust the decommissioning timeout threshold. Reduce the `spark.blacklist.decommissioning.timeout` from one hour to one minute to make the node available for other pending containers to continue task processing.

  You should also set `yarn.resourcemanager.nodemanager-graceful-decommission-timeout-secs` to a larger value to ensure Amazon EMR doesn't force terminate the node while the longest "Spark Task" is still running on the node. The current Default is 60 minutes, which means Yarn force terminates the container after 60 minutes once the node enters the decomissioning state.

  Here is a sample Yarn Resource Manager Log line showing nodes are added to the decomissioning state.

```
2021-10-20 15:55:26,994 INFO
 org.apache.hadoop.yarn.server.resourcemanager.DefaultAMSProcessor (IPC Server handler
 37 on default port 8030): blacklist are updated in Scheduler.blacklistAdditions:
 [ip-10-10-27-207.us-west-2.compute.internal, ip-10-10-29-216.us-west-2.compute.internal,
 ip-10-10-31-13.us-west-2.compute.internal, ... , ip-10-10-30-77.us-
west-2.compute.internal], blacklistRemovals: []
```

  More details on how Amazon EMR integrates with YARN deny listing during decommissioning of nodes.

  Cases when nodes in Amazon EMR can be deny listed.

  Configuring Spark node decommissioning behavior.
- Over-utilization of EBS volumes can cause Managed Scaling issues. We recommend that you maintain EBS volume below 90% utilization. See Specifying additional EBS storage volumes.

- EMR managed scaling is currently available in 22 AWS Regions: US East (N. Virginia and Ohio), US West (Oregon and N. California), South America (São Paulo), Europe (Frankfurt, Ireland, London, Milan, Paris, and Stockholm), Canada (Central), Asia Pacific (Hong Kong, Mumbai, Seoul, Singapore, Sydney, and Tokyo), Middle East (Bahrain), Africa (Cape Town), AWS GovCloud (US-East), AWS GovCloud (US-West), China (Beijing) operated by Sinnet, and China (Ningxia) operated by NWCD.
- Amazon CloudWatch metrics are critical for Amazon EMR managed scaling to operate. We recommend that you closely monitor Amazon CloudWatch metrics to make sure data is not missing. For more information about how you can configure CloudWatch alarms to detect missing metrics, see Using Amazon CloudWatch alarms.
- Managed scaling operations on 5.30.0 and 5.30.1 clusters without Presto installed may cause application failures or cause a uniform instance group or instance fleet to stay in the `ARRESTED` state, particularly when a scale down operation is followed quickly by a scale up operation.

  As a workaround, choose Presto as an application to install when you create a cluster with Amazon EMR releases 5.30.0 and 5.30.1, even if your job does not require Presto.
- EMR managed scaling only works with YARN applications, such as Spark, Hadoop, Hive, and Flink. It currently does not support applications that are not based on YARN, such as Presto and HBase.
- When you set the maximum core node and the On-Demand limit for EMR managed scaling, consider the differences between instance groups and instance fleets. Each instance group consists of the same instance type and the same purchasing option for instances: On-Demand or Spot. For each instance fleet, you can specify up to five instance types, which can be provisioned as On-Demand and Spot Instances. For more information, see Create a cluster with instance fleets or uniform instance groups, Instance fleet options, and Node allocation scenarios (p. 614).
- (EMR 5.30.0 and later) If you remove the default **Allow All** outbound rule to 0.0.0.0/ for the master security group, you must add a rule that allows outbound TCP connectivity to your security group for service access on port 9443. Your security group for service access must also allow inbound TCP traffic on port 9443 from the master security group. For more information about configuring security groups, see Amazon EMR-managed security group for the master instance (private subnets).
- You can use AWS CloudFormation to configure EMR managed scaling. For more information, see AWS::EMR::Cluster in the *AWS CloudFormation User Guide*.
- Managed scaling currently doesn't support the YARN node labels feature. Avoid using node labels on clusters with managed scaling. For example, don't allow executors to run only on task nodes. When you use node labels in your Amazon EMR clusters, you may find that your cluster isn't scaling up, which can lead to a slow-down of your application.

Topics

- Understanding node allocation strategy and scenarios (p. 613)
- Understanding managed scaling metrics (p. 617)
- Use the AWS Management Console to configure managed scaling (p. 621)
- Using the AWS CLI to configure managed scaling (p. 622)
- Using AWS SDK for Java to configure managed scaling (p. 623)

Understanding node allocation strategy and scenarios

This section gives an overview of node allocation strategy and common scaling scenarios that you can use with EMR managed scaling.

Node allocation strategy

EMR managed scaling allocates core and task nodes based on the following scale-up and scale-down strategies:

Scale-up strategy

- EMR managed scaling first adds capacity to core nodes and then to task nodes until the maximum allowed capacity is reached or until the desired scale-up target capacity is achieved.
- If the `MaximumCoreCapacityUnits` parameter is set, then Amazon EMR scales core nodes until the core units reach the maximum allowed limit. All the remaining capacity is added to task nodes.
- If the `MaximumOnDemandCapacityUnits` parameter is set, then Amazon EMR scales the cluster by using the On-Demand Instances until the On-Demand units reach the maximum allowed limit. All the remaining capacity is added using Spot Instances.
- If both the `MaximumCoreCapacityUnits` and `MaximumOnDemandCapacityUnits` parameters are set, Amazon EMR considers both limits during scaling.

  For example, if the `MaximumCoreCapacityUnits` is less than `MaximumOnDemandCapacityUnits`, EMR first scales core nodes until the core capacity limit is reached. For the remaining capacity, EMR first uses On-Demand Instances to scale task nodes until the On-Demand limit is reached, and then uses Spot Instances for task nodes.

Scale-down strategy

- Amazon EMR versions 5.34.0 and later, and Amazon EMR versions 6.4.0 and later, support managed scaling that is aware of Spark shuffle data (data that Spark redistributes across partitions to perform specific operations). For more information on shuffle operations, see the Spark Programming Guide. Managed scaling scales-down only instances that are under-utilized and which do not contain actively used shuffle data. This intelligent scaling prevents unintended shuffle data loss, avoiding the need for job re-attempts and recomputation of intermediate data.
- EMR managed scaling first removes task nodes and then removes core nodes until the desired scale-down target capacity is achieved. The cluster never scales below the minimum constraints in the managed scaling policy.
- Within each node type (either core nodes or task nodes), EMR managed scaling removes Spot Instances first and then removes On-Demand Instances.

If the cluster does not have any load, then Amazon EMR cancels the addition of new instances from a previous evaluation and performs scale-down operations. If the cluster has a heavy load, Amazon EMR cancels the removal of instances and performs scale-up operations.

Node allocation considerations

We recommend that you use the On-Demand purchasing option for core nodes to avoid HDFS data loss in case of Spot reclamation. You can use the Spot purchasing option for task nodes to reduce costs and get faster job execution when more Spot Instances are added to task nodes.

Node allocation scenarios

You can create various scaling scenarios based on your needs by setting up the Maximum, Minimum, On-Demand limit, and Maximum core node parameters in different combinations.

Scenario 1: Scale Core Nodes Only

To scale core nodes only, the managed scaling parameters must meet the following requirements:

- The On-Demand limit is equal to the maximum boundary.
- The maximum core node is equal to the maximum boundary.

When the On-Demand limit and the maximum core node parameters are not specified, both parameters default to the maximum boundary.

The following examples demonstrate the scenario of scaling cores nodes only.

Cluster initial state	Scaling parameters	Scaling behavior
Instance groups Core: 1 On-Demand Task: 1 On-Demand and 1 Spot	`UnitType`: Instances `MinimumCapacityUnits`: 1 `MaximumCapacityUnits`: 20 `MaximumOnDemandCapacityUnits`: 20 `MaximumCoreCapacityUnits`: 20	Scale between 1 to 20 Instances or instance fleet units on core nodes using On-Demand type. No scaling on task nodes.
Instance fleets Core: 1 On-Demand Task: 1 On-Demand and 1 Spot	UnitType: InstanceFleetUnits `MinimumCapacityUnits`: 1 `MaximumCapacityUnits`: 20 `MaximumOnDemandCapacityUnits`: 20 `MaximumCoreCapacityUnits`: 20	

Scenario 2: Scale task nodes only

To scale task nodes only, the managed scaling parameters must meet the following requirement:

- The maximum core node must be equal to the minimum boundary.

The following examples demonstrate the scenario of scaling task nodes only.

Cluster initial state	Scaling parameters	Scaling behavior
Instance groups Core: 2 On-Demand Task: 1 Spot	`UnitType`: Instances `MinimumCapacityUnits`: 2 `MaximumCapacityUnits`: 20 `MaximumCoreCapacityUnits`: 2	Keep core nodes steady at 2 and only scale task nodes between 0 to 18 instances or instance fleet units. The capacity between minimum and maximum boundaries is added to the task nodes only.
Instance fleets Core: 2 On-Demand Task: 1 Spot	`UnitType`: InstanceFleetUnits `MinimumCapacityUnits`: 2 `MaximumCapacityUnits`: 20 `MaximumCoreCapacityUnits`: 2	

Scenario 3: Only On-Demand Instances in the cluster

To have On-Demand Instances only, your cluster and the managed scaling parameters must meet the following requirement:

- The On-Demand limit is equal to the maximum boundary.

  When the On-Demand limit is not specified, the parameter value defaults to the maximum boundary. The default value indicates that Amazon EMR scales On-Demand Instances only.

If the maximum core node is less than the maximum boundary, the maximum core node parameter can be used to split capacity allocation between core and task nodes.

To enable this scenario in a cluster composed of instance groups, all node groups in the cluster must use the On-Demand market type during initial configuration.

The following examples demonstrate the scenario of having On-Demand Instances in the entire cluster.

Cluster initial state	Scaling parameters	Scaling behavior
Instance groups Core: 1 On-Demand Task: 1 On-Demand	`UnitType`: Instances `MinimumCapacityUnits`: 1 `MaximumCapacityUnits`: 20 `MaximumOnDemandCapacityUnits`: 20 `MaximumCoreCapacityUnits`: 12	Scale between 1 to 12 instances or instance fleet units on core nodes using On-Demand type. Scale the remaining capacity using On-Demand on task nodes. No scaling using Spot Instances.
Instance fleets Core: 1 On-Demand Task: 1 On-Demand	`UnitType`: InstanceFleetUnits `MinimumCapacityUnits`: 1 `MaximumCapacityUnits`: 20 `MaximumOnDemandCapacityUnits`: 20 `MaximumCoreCapacityUnits`: 12	

Scenario 4: Only Spot Instances in the cluster

To have Spot Instances only, the managed scaling parameters must meet the following requirement:

- On-Demand limit is set to 0.

If the maximum core node is less than the maximum boundary, the maximum core node parameter can be used to split capacity allocation between core and task nodes.

To enable this scenario in a cluster composed of instance groups, the core instance group must use the Spot purchasing option during initial configuration. If there is no Spot Instance in the task instance group, EMR managed scaling creates a task group using Spot Instances when needed.

The following examples demonstrate the scenario of having Spot Instances in the entire cluster.

Cluster initial state	Scaling parameters	Scaling behavior
Instance groups Core: 1 Spot Task: 1 Spot	`UnitType`: Instances `MinimumCapacityUnits`: 1 `MaximumCapacityUnits`: 20 `MaximumOnDemandCapacityUnits`: 0	Scale between 1 to 20 instances or instance fleet units on core nodes using Spot. No scaling using On-Demand type.
Instance fleets Core: 1 Spot	`UnitType`: InstanceFleetUnits `MinimumCapacityUnits`: 1	

Cluster initial state	Scaling parameters	Scaling behavior
Task: 1 Spot	`MaximumCapacityUnits`: 20 `MaximumOnDemandCapacityUnits`: 0	

Scenario 5: Scale On-Demand Instances on core nodes and Spot Instances on task nodes

To scale On-Demand Instances on core nodes and Spot Instances on task nodes, the managed scaling parameters must meet the following requirements:

- The On-Demand limit must be equal to the maximum core node.
- Both the On-Demand limit and the maximum core node must be less than the maximum boundary.

To enable this scenario in a cluster composed of instance groups, the core node group must use the On-Demand purchasing option.

The following examples demonstrate the scenario of scaling On-Demand Instances on core nodes and Spot Instances on task nodes.

Cluster initial state	Scaling parameters	Scaling behavior
Instance groups Core: 1 On-Demand Task: 1 On-Demand and 1 Spot	`UnitType`: Instances `MinimumCapacityUnits`: 1 `MaximumCapacityUnits`: 20 `MaximumOnDemandCapacityUnits`: 7 `MaximumCoreCapacityUnits`: 7	Scale up to 6 On-Demand units on the core node since there is already 1 On-Demand unit on the task node and the maximum limit for On-Demand is 7. Then scale up to 13 Spot units on task nodes.
Instance fleets Core: 1 On-Demand Task: 1 On-Demand and 1 Spot	`UnitType`: InstanceFleetUnits `MinimumCapacityUnits`: 1 `MaximumCapacityUnits`: 20 `MaximumOnDemandCapacityUnits`: 7 `MaximumCoreCapacityUnits`: 7	

Understanding managed scaling metrics

Amazon EMR publishes high-resolution metrics with data at a one-minute granularity when managed scaling is enabled for a cluster. You can view events on every resize initiation and completion controlled by managed scaling using the Amazon EMR console or the Amazon CloudWatch console. CloudWatch metrics are critical for Amazon EMR managed scaling to operate. We recommend that you closely monitor CloudWatch metrics to make sure data is not missing. For more information about how you can configure CloudWatch alarms to detect missing metrics, see Using Amazon CloudWatch alarms. For more information about using CloudWatch events with Amazon EMR, see Monitor CloudWatch events.

The following metrics indicate the current or target capacities of a cluster. These metrics are only available when managed scaling is enabled. For clusters composed of instance fleets, the cluster capacity metrics are measured in `Units`. For clusters composed of instance groups, the cluster capacity metrics are measured in `Nodes` or `vCPU` based on the unit type used in the managed scaling policy.

Metric	Description
• `TotalUnitsRequested` • `TotalNodesRequested` • `TotalVCPURequested`	The target total number of units/nodes/vCPUs in a cluster as determined by managed scaling. Units: *Count*
• `TotalUnitsRunning` • `TotalNodesRunning` • `TotalVCPURunning`	The current total number of units/nodes/vCPUs available in a running cluster. When a cluster resize is requested, this metric will be updated after the new instances are added or removed from the cluster. Units: *Count*
• `CoreUnitsRequested` • `CoreNodesRequested` • `CoreVCPURequested`	The target number of CORE units/nodes/vCPUs in a cluster as determined by managed scaling. Units: *Count*
• `CoreUnitsRunning` • `CoreNodesRunning` • `CoreVCPURunning`	The current number of CORE units/nodes/vCPUs running in a cluster. Units: *Count*
• `TaskUnitsRequested` • `TaskNodesRequested` • `TaskVCPURequested`	The target number of TASK units/nodes/vCPUs in a cluster as determined by managed scaling. Units: *Count*
• `TaskUnitsRunning` • `TaskNodesRunning` • `TaskVCPURunning`	The current number of TASK units/nodes/vCPUs running in a cluster. Units: *Count*

The following metrics indicate the usage status of cluster and applications. These metrics are available for all Amazon EMR features, but are published at a higher resolution with data at a one-minute granularity when managed scaling is enabled for a cluster. You can correlate the following metrics with the cluster capacity metrics in the previous table to understand the managed scaling decisions.

Metric	Description
`AppsCompleted`	The number of applications submitted to YARN that have completed. Use case: Monitor cluster progress Units: *Count*
`AppsPending`	The number of applications submitted to YARN that are in a pending state. Use case: Monitor cluster progress Units: *Count*
`AppsRunning`	The number of applications submitted to YARN that are running. Use case: Monitor cluster progress

Metric	Description
	Units: *Count*
`ContainerAllocated`	The number of resource containers allocated by the ResourceManager. Use case: Monitor cluster progress Units: *Count*
`ContainerPending`	The number of containers in the queue that have not yet been allocated. Use case: Monitor cluster progress Units: *Count*
`ContainerPendingRatio`	The ratio of pending containers to containers allocated (ContainerPendingRatio = ContainerPending / ContainerAllocated). If ContainerAllocated = 0, then ContainerPendingRatio = ContainerPending. The value of ContainerPendingRatio represents a number, not a percentage. This value is useful for scaling cluster resources based on container allocation behavior. Units: *Count*
`HDFSUtilization`	The percentage of HDFS storage currently used. Use case: Analyze cluster performance Units: *Percent*
`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 the entire five minutes. To avoid false positives, you should raise an alarm when this value has been 1 for more than one consecutive five-minute check. For example, you might raise an alarm on this value if it has been 1 for thirty minutes or longer. Use case: Monitor cluster performance Units: *Boolean*
`MemoryAvailableMB`	The amount of memory available to be allocated. Use case: Monitor cluster progress Units: *Count*

Metric	Description
`MRActiveNodes`	The number of nodes presently running MapReduce tasks or jobs. Equivalent to YARN metric `mapred.resourcemanager.NoOfActiveNodes`. Use case: Monitor cluster progress Units: *Count*
`YARNMemoryAvailablePercentage`	The percentage of remaining memory available to YARN (YARNMemoryAvailablePercentage = MemoryAvailableMB / MemoryTotalMB). This value is useful for scaling cluster resources based on YARN memory usage. Units: *Percent*

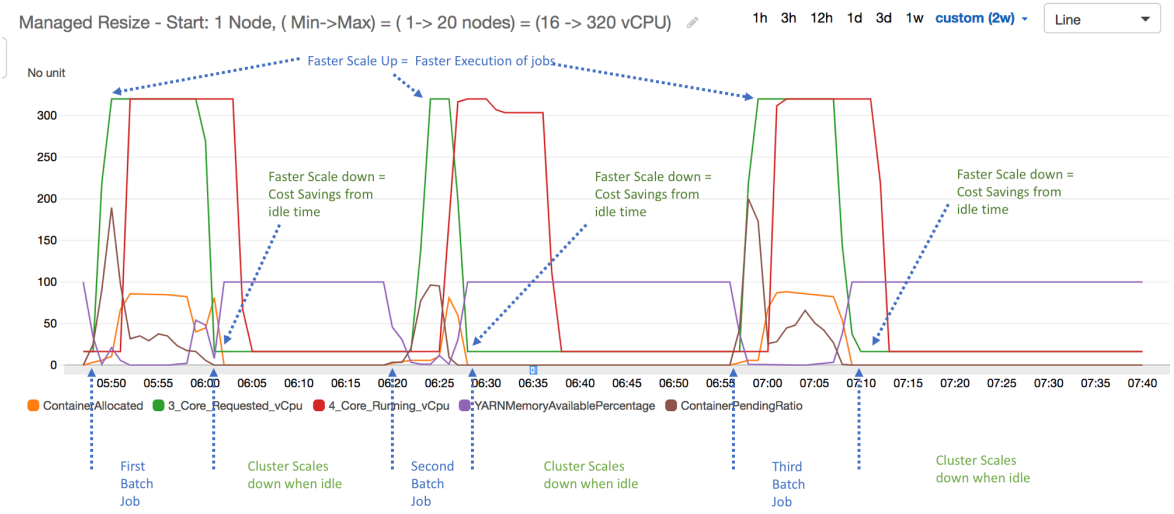
Graphing managed scaling metrics

You can graph metrics to visualize your cluster's workload patterns and corresponding scaling decisions made by EMR managed scaling as the following steps demonstrate.

To graph managed scaling metrics in the CloudWatch console

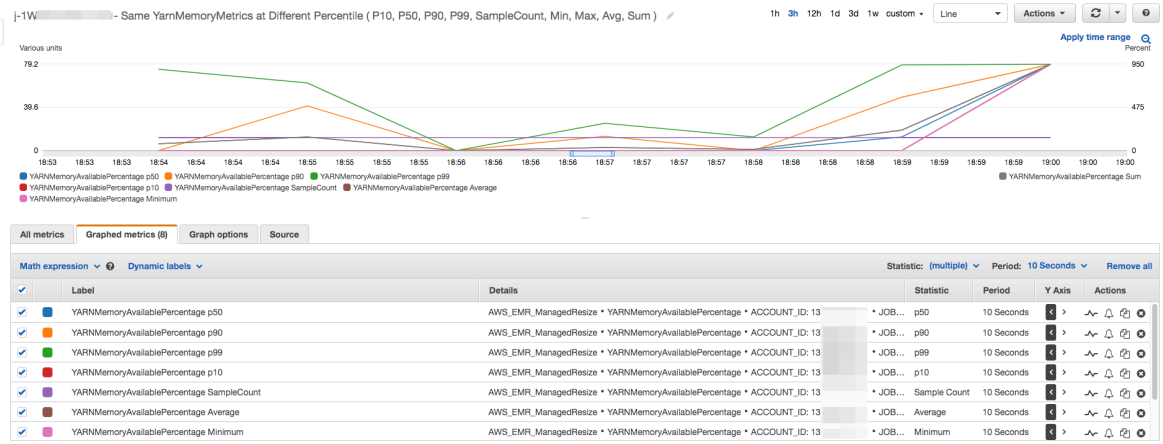
1. Open the CloudWatch console.
2. In the navigation pane, choose **Amazon EMR**. You can search on the cluster identifier of the cluster to monitor.
3. Scroll down to the metric to graph. Open a metric to display the graph.
4. To graph one or more metrics, select the check box next to each metric.

The following example illustrates the EMR managed scaling activity of a cluster. The graph shows three automatic scale-down periods, which save costs when there is a less active workload.



All the cluster capacity and usage metrics are published at one-minute intervals. Additional statistical information is also associated with each one-minute data, which allows you to plot various functions such as `Percentiles`, `Min`, `Max`, `Sum`, `Average`, `SampleCount`.

For example, the following graph plots the same `YARNMemoryAvailablePercentage` metric at different percentiles, P10, P50, P90, P99, along with `Sum`, `Average`, `Min`, `SampleCount`.



Use the AWS Management Console to configure managed scaling

You can use the Amazon EMR console to configure managed scaling when you create a cluster or to change a managed scaling policy for a running cluster.

When you create a cluster on the console, you can configure managed scaling with either quick options or advanced cluster configuration options. You can also create or change a managed scaling policy for a running cluster by modifying the **Managed Scaling** settings on the **Summary** or **Hardware** page.

To use quick options to configure managed scaling when you create a cluster with the console

1. Open the Amazon EMR console, choose **Create cluster** and open **Create Cluster - Quick options**.
2. In the **Hardware configuration** section next to **Cluster scaling**, choose the checkbox to enable **scale cluster nodes based on workload**.
3. Under **Core and task units**, specify the **Minimum** and **Maximum** number of core and task instances.

To use the advanced option to configure managed scaling when you create a cluster with the console

1. 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**.
2. In the **Cluster composition** section, select **Instance fleets** or **Uniform instance groups**.
3. Under **Cluster scaling**, select **Enable cluster scaling**. Then select **Use EMR-managed scaling**. Under **Core and task units**, specify the **Minimum** and **Maximum** number of instances or instance fleet units, the **On-Demand limit**, and **Maximum Core Node** count.

   For clusters composed of instance groups, you can also choose **Create a custom automatic scaling policy** if you want to define custom automatic scaling policies for each instance group. For more information, see Using automatic scaling with a custom policy for instance groups (p. 625).

To modify managed scaling on an existing cluster with the console

1. Open the Amazon EMR console, select your cluster from the cluster list, and then choose the **Hardware** tab.
2. In the **Cluster scaling** section, select **Edit** for EMR managed scaling.
3. In the **Cluster scaling** section, specify new values for the **Minimum** and **Maximum** number of instances and the **On-Demand limit**.

Using the AWS CLI to configure managed scaling

You can use AWS CLI commands for Amazon EMR to configure managed scaling when you create a cluster. 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 a managed scaling policy to an existing cluster and remove a managed scaling policy that was previously applied. In addition, you can retrieve details of a scaling policy configuration from a running cluster.

Enabling Managed Scaling During Cluster Launch

You can enable managed scaling during cluster launch as the following example demonstrates.

```
aws emr create-cluster \
 --service-role EMR_DefaultRole \
 --release-label emr-5.36.0 \
 --name EMR_Managed_Scaling_Enabled_Cluster \
 --applications Name=Spark Name=Hbase \
 --ec2-attributes KeyName=keyName,InstanceProfile=EMR_EC2_DefaultRole \
 --instance-groups InstanceType=m4.xlarge,InstanceGroupType=MASTER,InstanceCount=1
 InstanceType=m4.xlarge,InstanceGroupType=CORE,InstanceCount=2 \
 --region us-east-1 \
 --managed-scaling-policy
 ComputeLimits='{MinimumCapacityUnits=2,MaximumCapacityUnits=4,UnitType=Instances}'
```

You can also specify a managed policy configuration using the --managed-scaling-policy option when you use `create-cluster`.

Applying a Managed Scaling Policy to an Existing Cluster

You can apply a managed scaling policy to an existing cluster as the following example demonstrates.

```
aws emr put-managed-scaling-policy
--cluster-id j-123456
--managed-scaling-policy ComputeLimits='{MinimumCapacityUnits=1,
MaximumCapacityUnits=10,  MaximumOnDemandCapacityUnits=10, UnitType=Instances}'
```

You can also apply a managed scaling policy to an existing cluster by using the `aws emr put-managed-scaling-policy` command. The following example uses a reference to a JSON file, `managedscaleconfig.json`, that specifies the managed scaling policy configuration.

```
aws emr put-managed-scaling-policy --cluster-id j-123456 --managed-scaling-policy file://./
managedscaleconfig.json
```

The following example shows the contents of the `managedscaleconfig.json` file, which defines the managed scaling policy.

```
{
```

```
    "ComputeLimits": {
        "UnitType": "Instances",
        "MinimumCapacityUnits": 1,
        "MaximumCapacityUnits": 10,
        "MaximumOnDemandCapacityUnits": 10
    }
}
```

Retrieving a Managed Scaling Policy Configuration

The `GetManagedScalingPolicy` command retrieves the policy configuration. For example, the following command retrieves the configuration for the cluster with a cluster ID of `j-123456`.

```
aws emr get-managed-scaling-policy --cluster-id j-123456
```

The command produces the following example output.

```
{
   "ManagedScalingPolicy": {
      "ComputeLimits": {
         "MinimumCapacityUnits": 1,
         "MaximumOnDemandCapacityUnits": 10,
         "MaximumCapacityUnits": 10,
         "UnitType": "Instances"
      }
   }
}
```

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

Removing Managed Scaling Policy

The `RemoveManagedScalingPolicy` command removes the policy configuration. For example, the following command removes the configuration for the cluster with a cluster ID of `j-123456`.

```
aws emr remove-managed-scaling-policy --cluster-id j-123456
```

Using AWS SDK for Java to configure managed scaling

The following program excerpt shows how to configure managed scaling using the AWS SDK for Java:

```
package com.amazonaws.emr.sample;

import java.util.ArrayList;
import java.util.List;

import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduce;
import com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceClientBuilder;
import com.amazonaws.services.elasticmapreduce.model.Application;
import com.amazonaws.services.elasticmapreduce.model.ComputeLimits;
import com.amazonaws.services.elasticmapreduce.model.ComputeLimitsUnitType;
```

```
import com.amazonaws.services.elasticmapreduce.model.InstanceGroupConfig;
import com.amazonaws.services.elasticmapreduce.model.JobFlowInstancesConfig;
import com.amazonaws.services.elasticmapreduce.model.ManagedScalingPolicy;
import com.amazonaws.services.elasticmapreduce.model.RunJobFlowRequest;
import com.amazonaws.services.elasticmapreduce.model.RunJobFlowResult;

public class CreateClusterWithManagedScalingWithIG {

 public static void main(String[] args) {
  AWSCredentials credentialsFromProfile = getCreadentials("AWS-Profile-Name-Here");

  /**
   * Create an EMR client using the credentials and region specified in order to create the
 cluster
   */
  AmazonElasticMapReduce emr = AmazonElasticMapReduceClientBuilder.standard()
   .withCredentials(new AWSStaticCredentialsProvider(credentialsFromProfile))
   .withRegion(Regions.US_EAST_1)
   .build();

  /**
   * Create Instance Groups - Master, Core, Task
   */
  InstanceGroupConfig instanceGroupConfigMaster = new InstanceGroupConfig()
    .withInstanceCount(1)
    .withInstanceRole("MASTER")
    .withInstanceType("m4.large")
    .withMarket("ON_DEMAND");

  InstanceGroupConfig instanceGroupConfigCore = new InstanceGroupConfig()
   .withInstanceCount(4)
   .withInstanceRole("CORE")
   .withInstanceType("m4.large")
   .withMarket("ON_DEMAND");

  InstanceGroupConfig instanceGroupConfigTask = new InstanceGroupConfig()
   .withInstanceCount(5)
   .withInstanceRole("TASK")
   .withInstanceType("m4.large")
   .withMarket("ON_DEMAND");

  List<InstanceGroupConfig> igConfigs = new ArrayList<>();
  igConfigs.add(instanceGroupConfigMaster);
  igConfigs.add(instanceGroupConfigCore);
  igConfigs.add(instanceGroupConfigTask);

        /**
         *  specify applications to be installed and configured when EMR creates the
 cluster
         */
  Application hive = new Application().withName("Hive");
  Application spark = new Application().withName("Spark");
  Application ganglia = new Application().withName("Ganglia");
  Application zeppelin = new Application().withName("Zeppelin");

  /**
   * Managed Scaling Configuration -
         * Using UnitType=Instances for clusters composed of instance groups
   *
         * Other options are:
         * UnitType = VCPU ( for clusters composed of instance groups)
         * UnitType = InstanceFleetUnits ( for clusters composed of instance fleets)
         **/
  ComputeLimits computeLimits = new ComputeLimits()
    .withMinimumCapacityUnits(1)
    .withMaximumCapacityUnits(20)
```

```
    .withUnitType(ComputeLimitsUnitType.Instances);

  ManagedScalingPolicy managedScalingPolicy = new ManagedScalingPolicy();
  managedScalingPolicy.setComputeLimits(computeLimits);

  // create the cluster with a managed scaling policy
  RunJobFlowRequest request = new RunJobFlowRequest()
          .withName("EMR_Managed_Scaling_TestCluster")
          .withReleaseLabel("emr-5.36.0")          // Specifies the EMR release version
 label, we recommend the latest release
          .withApplications(hive,spark,ganglia,zeppelin)
          .withLogUri("s3://path/to/my/emr/logs")  // A URI in S3 for log files is required
 when debugging is enabled.
          .withServiceRole("EMR_DefaultRole")      // If you use a custom IAM service role,
 replace the default role with the custom role.
          .withJobFlowRole("EMR_EC2_DefaultRole")  // If you use a custom EMR role for EC2
 instance profile, replace the default role with the custom EMR role.
          .withInstances(new JobFlowInstancesConfig().withInstanceGroups(igConfigs)
              .withEc2SubnetId("subnet-123456789012345")
              .withEc2KeyName("my-ec2-key-name")
              .withKeepJobFlowAliveWhenNoSteps(true))
          .withManagedScalingPolicy(managedScalingPolicy);
    RunJobFlowResult result = emr.runJobFlow(request);

    System.out.println("The cluster ID is " + result.toString());
 }

 public static AWSCredentials getCredentials(String profileName) {
  // specifies any named profile in .aws/credentials as the credentials provider
  try {
   return new ProfileCredentialsProvider("AWS-Profile-Name-Here")
     .getCredentials();
        } catch (Exception e) {
            throw new AmazonClientException(
                    "Cannot load credentials from .aws/credentials file. " +
                    "Make sure that the credentials file exists and that the profile name
 is defined within it.",
                    e);
        }
 }

 public CreateClusterWithManagedScalingWithIG() { }
}
```

Using automatic scaling with a custom policy for instance groups

Topics

- Creating the IAM role for automatic scaling (p. 626)
- Understanding automatic scaling rules (p. 626)
- Considerations and limitations (p. 627)
- Using the AWS Management Console to configure automatic scaling (p. 628)
- Using the AWS CLI to configure automatic scaling (p. 628)

Automatic scaling with a custom policy 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 with a custom policy 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. 315).

> **Note**
> To use the automatic scaling with a custom policy feature in Amazon EMR, you must set `true` for the `VisibleToAllUsers` parameter when you create a cluster. For more information, see SetVisibleToAllUsers.

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 service roles for Amazon EMR permissions to AWS services and resources (p. 394).

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, Apache Hive, and Presto 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

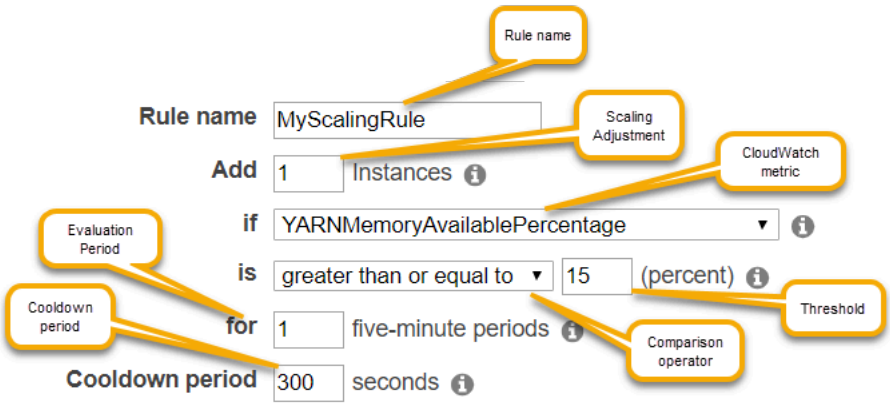
the lifecycle of Amazon EC2 instances that scale automatically, see Auto Scaling lifecycle in the *Amazon EC2 Auto Scaling User Guide*.

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. 640).

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 seconds, 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*.



Considerations and limitations

- Amazon CloudWatch metrics are critical for Amazon EMR automatic scaling to operate. We recommend that you closely monitor Amazon CloudWatch metrics to make sure data is not missing. For more information about how you can configure Amazon CloudWatch alarms to detect missing metrics, see Using Amazon CloudWatch alarms.

- Over-utilization of EBS volumes can cause Managed Scaling issues. We recommend that you monitor EBS volume usage closely to make sure EBS volume is below 90% utilization. See Instance storage for information on specifying additional EBS volumes.
- Automatic scaling with a custom policy in Amazon EMR release versions 5.18 to 5.28 may experience scaling failure caused by data intermittently missing in Amazon CloudWatch metrics. We recommend that you use the most recent Amazon EMR versions for improved autoscaling. You can also contact AWS Support for a patch if you wish to use an Amazon EMR release version between 5.18 and 5.28.

Using the AWS Management Console to configure automatic scaling

When you create a cluster, you configure a scaling policy for instance groups using the advanced cluster configuration options. You can also create or modify a scaling policy for an instance group in-service by modifying instance groups in the **Hardware** settings of an existing cluster.

1. If you are creating a cluster, in the Amazon EMR console, select **Create Cluster**, select **Go to advanced options**, choose options for **Step 1: Software and Steps**, and then go to **Step 2: Hardware Configuration**.

   - or -

   If you are modifying an instance group in a running cluster, select your cluster from the cluster list, and then expand the **Hardware** section.
2. In **Cluster scaling** section, select **Enable cluster scaling**. Then select **Create a custom automatic scaling policy**.

   In the table of **Custom automatic scaling policies**, click the pencil icon that appears in the row of the instance group you want to configure. The Auto Scaling Rules screen opens.
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 automatic scaling. 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=m5.xlarge,InstanceCount=1
 'Name=MyCoreIG,InstanceGroupType=CORE,InstanceType=m5.xlarge,InstanceCount=2,AutoScalingPolicy={Const
scale-out,Description=Replicates the default scale-out rule in the
 console.,Action={SimpleScalingPolicyConfiguration={AdjustmentType=CHANGE_IN_CAPACITY,ScalingAdjustment
ElasticMapReduce,Period=300,Statistic=AVERAGE,Threshold=15,Unit=PERCENT,Dimensions=[{Key=JobFlowId,Valu
```

The following command illustrates using the command line to provide the automatic scaling policy definition as part of an instance group configuration file named *`instancegroupconfig.json`*.

```
aws emr create-cluster --release-label emr-5.2.0 --service-role EMR_DefaultRole --ec2-
attributes InstanceProfile=EMR_EC2_DefaultRole --instance-groups file:///your/path/to/
instancegroupconfig.json --auto-scaling-role EMR_AutoScaling_DefaultRole
```

With the contents of the configuration file as follows:

```
[
{
  "InstanceCount": 1,
  "Name": "MyMasterIG",
  "InstanceGroupType": "MASTER",
  "InstanceType": "m5.xlarge"
},
{
  "InstanceCount": 2,
  "Name": "MyCoreIG",
  "InstanceGroupType": "CORE",
  "InstanceType": "m5.xlarge",
  "AutoScalingPolicy":
    {
     "Constraints":
      {
       "MinCapacity": 2,
       "MaxCapacity": 10
      },
     "Rules":
     [
      {
       "Name": "Default-scale-out",
       "Description": "Replicates the default scale-out rule in the console for YARN
 memory.",
       "Action":{
        "SimpleScalingPolicyConfiguration":{
          "AdjustmentType": "CHANGE_IN_CAPACITY",
          "ScalingAdjustment": 1,
```

```
          "CoolDown": 300
        }
       },
       "Trigger":{
        "CloudWatchAlarmDefinition":{
          "ComparisonOperator": "LESS_THAN",
          "EvaluationPeriods": 1,
          "MetricName": "YARNMemoryAvailablePercentage",
          "Namespace": "AWS/ElasticMapReduce",
          "Period": 300,
          "Threshold": 15,
          "Statistic": "AVERAGE",
          "Unit": "PERCENT",
          "Dimensions":[
             {
               "Key" : "JobFlowId",
               "Value" : "${emr.clusterId}"
             }
          ]
        }
       }
      }
     ]
   }
}
]
```

Adding an instance group with an automatic scaling policy to a cluster

You can specify a scaling policy configuration using the `--instance-groups` option with the `add-instance-groups` command in the same way you can when you use `create-cluster`. The following example uses a reference to a JSON file, *`instancegroupconfig.json`*, with the instance group configuration.

```
aws emr add-instance-groups --cluster-id j-1EKZ3TYEVF1S2 --instance-groups file://your/
path/to/instancegroupconfig.json
```

Applying an automatic scaling policy to an existing instance group or modifying an applied policy

Use the `aws emr put-auto-scaling-policy` command to apply an automatic scaling policy to an existing instance group. The instance group must be part of a cluster that uses the automatic scaling IAM role. The following example uses a reference to a JSON file, *`autoscaleconfig.json`*, that specifies the automatic scaling policy configuration.

```
aws emr put-auto-scaling-policy --cluster-id j-1EKZ3TYEVF1S2 --instance-group-
id ig-3PLUZBA6WLS07 --auto-scaling-policy file://your/path/to/autoscaleconfig.json
```

The contents of the `autoscaleconfig.json` file, which defines the same scale-out rule as shown in the previous example, is shown below.

```
{
          "Constraints": {
                  "MaxCapacity": 10,
                  "MinCapacity": 2
          },
```

```
        "Rules": [{
                "Action": {
                        "SimpleScalingPolicyConfiguration": {
                                "AdjustmentType": "CHANGE_IN_CAPACITY",
                                "CoolDown": 300,
                                "ScalingAdjustment": 1
                        }
                },
                "Description": "Replicates the default scale-out rule in the console for
 YARN memory",
                "Name": "Default-scale-out",
                "Trigger": {
                        "CloudWatchAlarmDefinition": {
                                "ComparisonOperator": "LESS_THAN",
                                "Dimensions": [{
                                        "Key": "JobFlowID",
                                        "Value": "${emr.clusterID}"
                                }],
                                "EvaluationPeriods": 1,
                                "MetricName": "YARNMemoryAvailablePercentage",
                                "Namespace": "AWS/ElasticMapReduce",
                                "Period": 300,
                                "Statistic": "AVERAGE",
                                "Threshold": 15,
                                "Unit": "PERCENT"
                        }
                }
        }]
 }
```

Removing an automatic scaling policy from an instance group

```
aws emr remove-auto-scaling-policy --cluster-id j-1EKZ3TYEVF1S2 --instance-group-
id ig-3PLUZBA6WLS07
```

Retrieving an automatic scaling policy configuration

The `describe-cluster` command retrieves the policy configuration in the InstanceGroup block. For example, the following command retrieves the configuration for the cluster with a cluster ID of `j-1CWOHP4PI30VJ`.

```
aws emr describe-cluster --cluster-id j-1CWOHP4PI30VJ
```

The command produces the following example output.

```
{
    "Cluster": {
        "Configurations": [],
        "Id": "j-1CWOHP4PI30VJ",
        "NormalizedInstanceHours": 48,
        "Name": "Auto Scaling Cluster",
        "ReleaseLabel": "emr-5.2.0",
        "ServiceRole": "EMR_DefaultRole",
        "AutoTerminate": false,
        "TerminationProtected": true,
        "MasterPublicDnsName": "ec2-54-167-31-38.compute-1.amazonaws.com",
        "LogUri": "s3n://aws-logs-232939870606-us-east-1/elasticmapreduce/",
```

```
        "Ec2InstanceAttributes": {
            "Ec2KeyName": "performance",
            "AdditionalMasterSecurityGroups": [],
            "AdditionalSlaveSecurityGroups": [],
            "EmrManagedSlaveSecurityGroup": "sg-09fc9362",
            "Ec2AvailabilityZone": "us-east-1d",
            "EmrManagedMasterSecurityGroup": "sg-0bfc9360",
            "IamInstanceProfile": "EMR_EC2_DefaultRole"
        },
        "Applications": [
            {
                "Name": "Hadoop",
                "Version": "2.7.3"
            }
        ],
        "InstanceGroups": [
            {
                "AutoScalingPolicy": {
                    "Status": {
                        "State": "ATTACHED",
                        "StateChangeReason": {
                            "Message": ""
                        }
                    },
                    "Constraints": {
                        "MaxCapacity": 10,
                        "MinCapacity": 2
                    },
                    "Rules": [
                        {
                            "Name": "Default-scale-out",
                            "Trigger": {
                                "CloudWatchAlarmDefinition": {
                                    "MetricName": "YARNMemoryAvailablePercentage",
                                    "Unit": "PERCENT",
                                    "Namespace": "AWS/ElasticMapReduce",
                                    "Threshold": 15,
                                    "Dimensions": [
                                        {
                                            "Key": "JobFlowId",
                                            "Value": "j-1CWOHP4PI30VJ"
                                        }
                                    ],
                                    "EvaluationPeriods": 1,
                                    "Period": 300,
                                    "ComparisonOperator": "LESS_THAN",
                                    "Statistic": "AVERAGE"
                                }
                            },
                            "Description": "",
                            "Action": {
                                "SimpleScalingPolicyConfiguration": {
                                    "CoolDown": 300,
                                    "AdjustmentType": "CHANGE_IN_CAPACITY",
                                    "ScalingAdjustment": 1
                                }
                            }
                        },
                        {
                            "Name": "Default-scale-in",
                            "Trigger": {
                                "CloudWatchAlarmDefinition": {
                                    "MetricName": "YARNMemoryAvailablePercentage",
                                    "Unit": "PERCENT",
                                    "Namespace": "AWS/ElasticMapReduce",
                                    "Threshold": 75,
```

```
                                "Dimensions": [
                                    {
                                        "Key": "JobFlowId",
                                        "Value": "j-1CWOHP4PI30VJ"
                                    }
                                ],
                                "EvaluationPeriods": 1,
                                "Period": 300,
                                "ComparisonOperator": "GREATER_THAN",
                                "Statistic": "AVERAGE"
                            }
                        },
                        "Description": "",
                        "Action": {
                            "SimpleScalingPolicyConfiguration": {
                                "CoolDown": 300,
                                "AdjustmentType": "CHANGE_IN_CAPACITY",
                                "ScalingAdjustment": -1
                            }
                        }
                    }
                ]
            },
            "Configurations": [],
            "InstanceType": "m5.xlarge",
            "Market": "ON_DEMAND",
            "Name": "Core - 2",
            "ShrinkPolicy": {},
            "Status": {
                "Timeline": {
                    "CreationDateTime": 1479413437.342,
                    "ReadyDateTime": 1479413864.615
                },
                "State": "RUNNING",
                "StateChangeReason": {
                    "Message": ""
                }
            },
            "RunningInstanceCount": 2,
            "Id": "ig-3M16XBE8C3PH1",
            "InstanceGroupType": "CORE",
            "RequestedInstanceCount": 2,
            "EbsBlockDevices": []
        },
        {
            "Configurations": [],
            "Id": "ig-OP62I28NSE8M",
            "InstanceGroupType": "MASTER",
            "InstanceType": "m5.xlarge",
            "Market": "ON_DEMAND",
            "Name": "Master - 1",
            "ShrinkPolicy": {},
            "EbsBlockDevices": [],
            "RequestedInstanceCount": 1,
            "Status": {
                "Timeline": {
                    "CreationDateTime": 1479413437.342,
                    "ReadyDateTime": 1479413752.088
                },
                "State": "RUNNING",
                "StateChangeReason": {
                    "Message": ""
                }
            },
            "RunningInstanceCount": 1
        }
```

```
        ],
        "AutoScalingRole": "EMR_AutoScaling_DefaultRole",
        "Tags": [],
        "BootstrapActions": [],
        "Status": {
            "Timeline": {
                "CreationDateTime": 1479413437.339,
                "ReadyDateTime": 1479413863.666
            },
            "State": "WAITING",
            "StateChangeReason": {
                "Message": "Cluster ready after last step completed."
            }
        }
    }
}
```

Manually resizing a running cluster

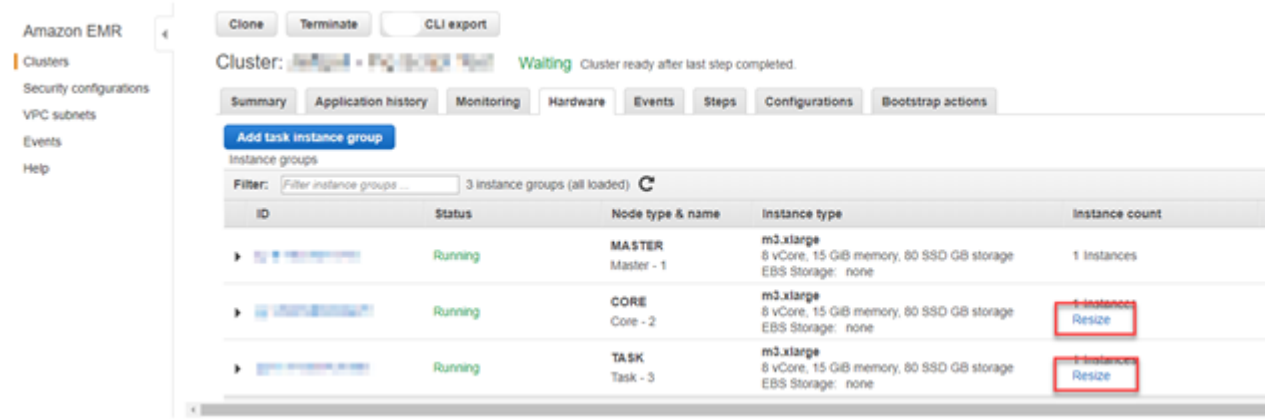
You can add and remove instances from core and task instance groups and instance fleets in a running cluster using the AWS Management Console, AWS CLI, or the Amazon EMR API. If a cluster uses instance groups, you explicitly change the instance count. If your cluster uses instance fleets, you can change 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. 317). 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. 640).

Resize a cluster using the console

You can use the Amazon EMR console to resize a running cluster.

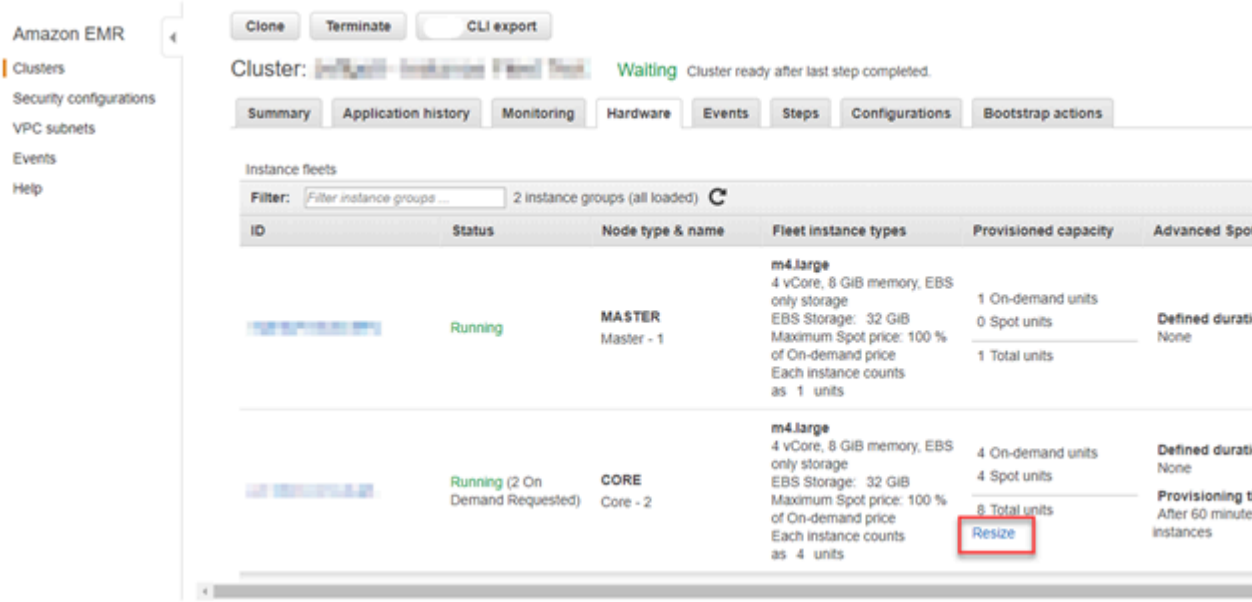
To change the instance count for an existing running cluster using the console

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.



–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**.



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-31JXXXXXXBT0`* with the instance group ID.

```
aws emr modify-instance-groups --instance-groups
 InstanceGroupId=ig-31JXXXXXXBT0,InstanceCount=4
```

  To retrieve the `InstanceGroupId`, use the `describe-cluster` subcommand. The output is a JSON object called `Cluster` that contains the ID of each instance group. To use this command, you need the cluster ID (which you can retrieve using the `aws emr list-clusters` command

or the console). To retrieve the instance group ID, type the following command and replace *j-2AXXXXXXGAPLF* with the cluster ID.

```
aws emr describe-cluster --cluster-id j-2AXXXXXXGAPLF
```

Using the AWS CLI, you can also terminate an instance in the core instance group with the `--modify-instance-groups` subcommand.

> **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-f9XXXXf2* with the instance ID.

```
aws emr modify-instance-groups --instance-groups
 InstanceGroupId=ig-6RXXXXXX07SA,EC2InstanceIdsToTerminate=i-f9XXXXf2
```

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=m5.xlarge
   ```

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=m5.xlarge
    InstanceCount=10,InstanceGroupType=task,InstanceType=m5.xlarge
   ```

   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-myInstanceGroupId,InstanceCount=10
```

  The second request to stop the first request:

```
aws emr modify-instance-groups --instance-groups
 InstanceGroupId=ig-myInstanceGroupId,InstanceCount=6
```

  Note
  Because this process is asynchronous, you may see instance counts change with respect to previous API requests before subsequent requests are honored. In the case of shrinking, it is possible that if you have work running on the nodes, the instance group may not shrink until nodes have completed their work.

Suspended state

An instance group goes into a suspended 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 a *SUSPENDED* 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 SUSPENDED.

To reset a cluster in a SUSPENDED 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-3KVXXXXXXY7UG* with the cluster ID.

```
aws emr describe-cluster --cluster-id j-3KVXXXXXXY7UG
```

  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": "m5.xlarge",
                "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": "m5.xlarge",
                "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 SUSPENDED 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

> **Note**
> Scale-down behavior options are no longer supported since Amazon EMR release version 5.10.0. Because of the introduction of per-second billing in Amazon EC2, the default scale-down behavior for Amazon EMR clusters is now terminate at task completion.

With Amazon EMR release versions 5.1.0 through 5.9.1, 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 deny lists 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 Graceful Decommission, 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.

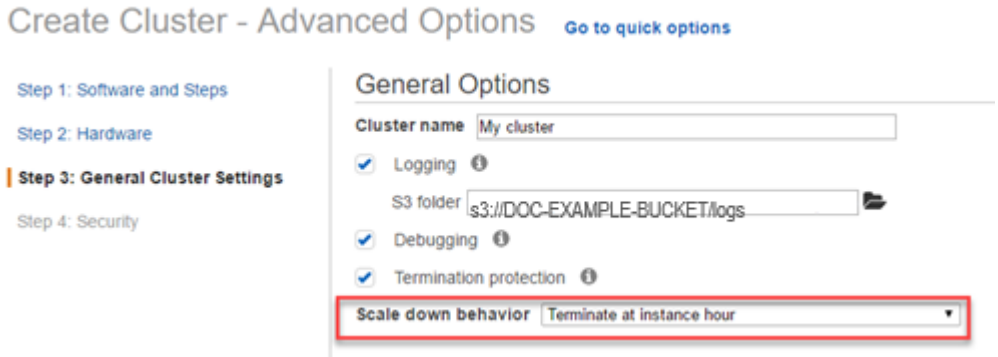
For HDFS, graceful shrink ensures that the target capacity of HDFS is large enough to fit all existing blocks. If the target capacity is not large enough, only a partial amount of core instances are decommissioned such that the remaining nodes can handle the current data residing in HDFS. You should 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.

Configure Amazon EMR scale-down behavior

> **Note**
> The terminate at instance hour scale-down behavior option is no longer supported for Amazon EMR release versions 5.10.0 and later. The following scale-down behavior options only appear in the Amazon EMR console for release versions 5.1.0 through 5.9.1.

You can use the AWS Management Console, the AWS CLI, or the Amazon EMR API to configure scale-down behavior when you create a cluster. Configuring scale-down using the AWS Management Console is done in the **Step 3: General Cluster Settings** screen when you create a cluster using **Advanced options**.



When you create a cluster using the AWS CLI, use the `--scale-down-behavior` 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. 642).

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 RUNNING 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 RUNNING or PENDING at any given time.

With Amazon EMR versions 4.8.0 and later, except version 5.0.0, you can cancel pending steps using the AWS Management Console, the AWS CLI, or the Amazon EMR API.

With Amazon EMR versions 5.28.0 and later, you can cancel both pending and running steps. You can also choose to run multiple steps in parallel to improve cluster utilization and save cost.

Note
For the best performance, we recommend that you store custom bootstrap actions, scripts, and other files that you want to use with Amazon EMR in an Amazon S3 bucket that is in the same AWS Region as your cluster.

Topics

- Adding steps to a cluster using the console (p. 642)
- Adding steps to a cluster using the AWS CLI (p. 644)
- Considerations for running multiple steps in parallel (p. 645)
- Viewing steps (p. 646)
- Canceling steps (p. 646)

Adding steps to a cluster using the console

Use the following procedures to add steps to a cluster using the AWS Management Console. For detailed information about how to submit steps for specific big data applications, see the *Amazon EMR Release Guide*.

Add steps during cluster creation

Using the AWS Management Console, you can add steps to a cluster when the cluster is created.

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/home. Choose **Create Cluster - Advanced Options**.
2. On the **Step 1: Software and Steps** page, for **Steps (optional)**, select **Run multiple steps in parallel to improve cluster utilization and save cost**. The default value for the concurrency level is 10. You can choose between 2 and 256 steps that can run in parallel.

Note
Running multiple steps in parallel is only supported with Amazon EMR version 5.28.0 and later.

3. For **After last step completes**, choose **Cluster enters waiting state** or **Auto-terminate the cluster**.
4. Choose **Step type**, then **Add step**.
5. Type appropriate values in the fields in the **Add Step** dialog. Options differ depending on the step type. If you have enabled **Run multiple steps in parallel to improve cluster utilization and save cost**, the only available option for **Action on failure** is **Continue**. Next, choose **Add**.

Add steps to a running cluster

Using the AWS Management Console, you can add steps to a long-running cluster - that is, a cluster with the auto-terminate option disabled.

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/home. On the **Cluster List** page, select the link for your cluster.
2. On the **Cluster Details** page, choose the **Steps** tab.
3. On the **Steps** tab, choose **Add step**.
4. Type appropriate values in the fields in the **Add Step** dialog, and then choose **Add**. The options differ depending on the step type.

Modify the step concurrency level in a running cluster

Using the AWS Management Console, you can modify the step concurrency level in a running cluster.

Note
Running multiple steps in parallel is only supported with Amazon EMR version 5.28.0 and later.

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/home. On the **Cluster List** page, select the link for your cluster.
2. On the **Cluster Details** page, choose the **Steps** tab.
3. For **Concurrency**, choose **Change**. Select a new value for the step concurrency level and then save.

Add step arguments

When you add a step to your cluster using the AWS Management Console, you can specify arguments for that step in the **Arguments** field. You must separate arguments with whitespace and surround string arguments that consist of characters *and* whitespace with quotation marks.

Example : Correct arguments

The following example arguments are formatted correctly for the AWS Management Console, with quotation marks around the final string argument.

```
bash -c "aws s3 cp s3://DOC-EXAMPLE-BUCKET/my-script.sh ."
```

You can also put each argument on a separate line for readability like the following example.

```
bash
-c
"aws s3 cp s3://DOC-EXAMPLE-BUCKET/my-script.sh ."
```

Example : Incorrect arguments

The following example arguments are improperly formatted for the AWS Management Console. Notice that the final string argument, `aws s3 cp s3://DOC-EXAMPLE-BUCKET/my-script.sh .`, contains whitespace and is not surrounded by quotation marks.

```
bash -c aws s3 cp s3://DOC-EXAMPLE-BUCKET/my-script.sh .
```

Adding steps to a cluster 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 steps during cluster creation

- Type the following command to create a cluster and add an Apache Pig step. Make sure to replace *myKey* with the name of your Amazon EC2 key pair.

```
aws emr create-cluster --name "Test cluster" \
--applications Name=Spark \
--use-default-roles \
--ec2-attributes KeyName=myKey \
--instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m5.xlarge
 InstanceGroupType=CORE,InstanceCount=2,InstanceType=m5.xlarge \
--steps '[{"Args":["spark-submit","--deploy-mode","cluster","--
class","org.apache.spark.examples.SparkPi","/usr/lib/spark/examples/jars/spark-
examples.jar","5"],"Type":"CUSTOM_JAR","ActionOnFailure":"CONTINUE","Jar":"command-
runner.jar","Properties":"","Name":"Spark application"}]'
```

  Note
  The list of arguments changes depending on the type of step.

  By default, the step concurrency level is 1. You can set the step concurrency level by using the `StepConcurrencyLevel` parameter when you create a cluster.

  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 own cluster ID.

```
aws emr add-steps --cluster-id j-2AXXXXXXGAPLF \
--steps '[{"Args":["spark-submit","--deploy-mode","cluster","--
class","org.apache.spark.examples.SparkPi","/usr/lib/spark/examples/jars/spark-
examples.jar","5"],"Type":"CUSTOM_JAR","ActionOnFailure":"CONTINUE","Jar":"command-
runner.jar","Properties":"","Name":"Spark application"}]'
```

  The output is a step identifier similar to the following.

```
{
```

```
    "StepIds": [
 "s-Y9XXXXXXAPMD"
    ]
}
```

To modify the StepConcurrencyLevel in a running cluster

1. In a running cluster, you can modify the StepConcurrencyLevel by using the `ModifyCluster` API. For example, type the following command to increase the StepConcurrenyLevel to `10`. Replace *`j-2AXXXXXXGAPLF`* with your cluster ID.

```
aws emr modify-cluster --cluster-id j-2AXXXXXXGAPLF --step-concurrency-level 10
```

2. The output is similar to the following.

```
{
"StepConcurrencyLevel": 10
}
```

For more information on using Amazon EMR commands in the AWS CLI, see the AWS CLI Command Reference.

Considerations for running multiple steps in parallel

- Steps running in parallel may complete in any order, but pending steps in queue transition to running state in the order they were submitted.
- When you select a step concurrency level for your cluster, you must consider whether or not the master node instance type meets the memory requirements of user workloads. The main step executer process runs on the master node for each step. Running multiple steps in parallel requires more memory and CPU utilization from the master node than running one step at a time.
- To achieve complex scheduling and resource management of concurrent steps, you can use YARN scheduling features such as `FairScheduler` or `CapacityScheduler`. For example, you can use `FairScheduler` with a `queueMaxAppsDefault` set to prevent more than a certain number of jobs from running at a time.
- The step concurrency level is subject to the configurations of resource managers. For example, if YARN is configured with only a parallelism of `5`, then you can only have five YARN applications running in parallel even if the `StepConcurrencyLevel` is set to `10`. For more information about configuring resource managers, see Configure applications in the *Amazon EMR Release Guide*.
- You cannot add a step with an `ActionOnFailure` other than CONTINUE while the step concurrency level of the cluster is greater than 1.
- If the step concurrency level of a cluster is greater than one, step `ActionOnFailure` feature will not activate.
- If a cluster has step concurrency level 1 but has multiple running steps, `TERMINATE_CLUSTER` `ActionOnFailure` may activate, but `CANCEL_AND_WAIT` `ActionOnFailure` will not. This edge case arises when the cluster step concurrency level was greater than one, but lowered while multiple steps were running.
- You can use EMR automatic scaling to scale up and down based on the YARN resources to prevent resource contention. For more information, see Using automatic scaling with a custom policy for instance groups in the *Amazon EMR Management Guide*.
- When you decrease the step concurrent level, EMR allows any running steps to complete before reducing the number of steps. If the resources are exhausted because the cluster is running too many concurrent steps, we recommend manually canceling any running steps to free up resources.

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 AWS 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 table indicates 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

Canceling steps

You can cancel pending and running steps using the AWS Management Console, the AWS CLI, or the Amazon EMR API.

To cancel steps using the AWS Management Console

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/home. 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**. Then choose **Cancel step**.
4. In the **Cancel step** dialog, keep the default option **Cancel the step and wait for it to exit**. If you want to end the step immediately without waiting for any processes to complete, choose **Cancel the step and force it to exit**.
5. Choose **Cancel 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-2QUAXXXXXXXXX \
--step-ids s-3M8DXXXXXXXXX s-3M8DXXXXXXXXX \
--step-cancellation-option SEND_INTERRUPT
```

With Amazon EMR version 5.28.0, you can choose one of the two following cancellation options for `StepCancellationOption` parameter when canceling steps.

- `SEND_INTERRUPT`– This is the default option. When a step cancellation request is received, EMR sends a `SIGTERM` signal to the step. add a `SIGTERM` signal handler to your step logic to catch this signal and terminate descendant step processes or wait for them to complete.
- `TERMINATE_PROCESS` – When this option is selected, EMR sends a `SIGKILL` signal to the step and all its descendant processes which terminates them immediately.

Considerations for canceling steps

- Canceling a running or pending step removes that step from the active step count.
- Canceling a running step does not allow a pending step to start running, assuming no change to `stepConcurrencyLevel`.
- Canceling a running step does not trigger the step `ActionOnFailure`.
- For EMR 5.32.0 and later, `SEND_INTERRUPT StepCancellationOption` sends a `SIGTERM` signal to the step child process. You should watch for this signal and do a cleanup and shutdown gracefully. The `TERMINATE_PROCESS StepCancellationOption` sends a `SIGKILL` signal to the step child process and all of its descendant processes; however, asynchronous processes are not affected.

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. 648)
- Viewing and restarting Amazon EMR and application processes (daemons) (p. 650)
- Troubleshoot a failed cluster (p. 653)
- Troubleshoot a slow cluster (p. 657)
- Common errors in Amazon EMR (p. 663)
- Troubleshoot a Lake Formation cluster (p. 679)

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. 342) and Step 5: Test the cluster step by step (p. 656).

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. 648)
- Tools to run scripts and configure processes (p. 649)
- Tools to view log files (p. 649)
- Tools to monitor cluster performance (p. 649)

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. 550).

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 user interfaces** 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. 557). 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 run scripts and configure processes

As part of your troubleshooting process, you might find it helpful to run custom scripts on your cluster or view and configure cluster processes.

View and restart application processes

It can be helpful to view running processes on your cluster in order to diagnose potential issues. You can stop and restart cluster processes by connecting to the master node of your cluster. For more information, see Viewing and restarting Amazon EMR and application processes (daemons) (p. 650).

Run commands and scripts without an SSH connection

To run a command or a script on your cluster as a step, you can use the `command-runner.jar` or `script-runner.jar` tools without establishing an SSH connection to the master node. For more information, see Run commands and scripts on an Amazon EMR cluster.

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. 342).

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. 599).

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. 577).

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, you can restart a process 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 must first connect to the master node. For more information, see Connect to the cluster (p. 591).

Viewing running processes

The method you use to view running processes on a cluster differs according to the Amazon EMR version you use.

EMR 5.30.0 and 6.0.0 and later

Example : List all running processes

The following example uses `systemctl` and specifies `--type` to view all processes.

```
systemctl --type=service
```

Example : List specific processes

The following example lists all processes with names that contain hadoop.

```
systemctl --type=service | grep -i hadoop
```

Example output:

```
hadoop-hdfs-namenode.service           loaded active running Hadoop namenode
hadoop-httpfs.service                  loaded active running Hadoop httpfs
hadoop-kms.service                     loaded active running Hadoop kms
```

```
 hadoop-mapreduce-historyserver.service loaded active running Hadoop historyserver
 hadoop-state-pusher.service            loaded active running Daemon process that
 processes and serves EMR metrics data.
 hadoop-yarn-proxyserver.service        loaded active running Hadoop proxyserver
 hadoop-yarn-resourcemanager.service    loaded active running Hadoop resourcemanager
 hadoop-yarn-timelineserver.service     loaded active running Hadoop timelineserver
```

Example : See a detailed status report for a specific process

The following example displays a detailed status report for the `hadoop-hdfs-namenode` service.

```
sudo systemctl status hadoop-hdfs-namenode
```

Example output:

```
hadoop-hdfs-namenode.service - Hadoop namenode
   Loaded: loaded (/etc/systemd/system/hadoop-hdfs-namenode.service; enabled; vendor
 preset: disabled)
   Active: active (running) since Wed 2021-08-18 21:01:46 UTC; 26min ago
 Main PID: 9733 (java)
    Tasks: 0
   Memory: 1.1M
   CGroup: /system.slice/hadoop-hdfs-namenode.service
           # 9733 /etc/alternatives/jre/bin/java -Dproc_namenode -Xmx1843m -server -
XX:OnOutOfMemoryError=kill -9 %p ...

Aug 18 21:01:37 ip-172-31-20-123 systemd[1]: Starting Hadoop namenode...
Aug 18 21:01:37 ip-172-31-20-123 su[9715]: (to hdfs) root on none
Aug 18 21:01:37 ip-172-31-20-123 hadoop-hdfs-namenode[9683]: starting namenode, logging
 to /var/log/hadoop-hdfs/ha...out
Aug 18 21:01:46 ip-172-31-20-123 hadoop-hdfs-namenode[9683]: Started Hadoop namenode:[
  OK  ]
Aug 18 21:01:46 ip-172-31-20-123 systemd[1]: Started Hadoop namenode.
Hint: Some lines were ellipsized, use -l to show in full.
```

EMR 4.x - 5.29.0

Example : List all running processes

The following example lists all running processes.

```
initctl list
```

EMR 2.x - 3.x

Example : List all running processes

The following example lists all running processes.

```
ls /etc/init.d/
```

Stopping and restarting processes

After you determine which processes are running, you can stop and then restart them if necessary.

EMR 5.30.0 and 6.0.0 and later

Example : Stop a process

The following example stops the `hadoop-hdfs-namenode` process.

```
sudo systemctl stop hadoop-hdfs-namenode
```

You can query the `status` to verify that the process is stopped.

```
sudo systemctl status hadoop-hdfs-namenode
```

Example output:

```
hadoop-hdfs-namenode.service - Hadoop namenode
  Loaded: loaded (/etc/systemd/system/hadoop-hdfs-namenode.service; enabled; vendor
 preset: disabled)
  Active: failed (Result: exit-code) since Wed 2021-08-18 21:37:50 UTC; 8s ago
Main PID: 9733 (code=exited, status=143)
```

Example : Start a process

The following example starts the `hadoop-hdfs-namenode` process.

```
sudo systemctl start hadoop-hdfs-namenode
```

You can query the status to verify that the process is running.

```
sudo systemctl status hadoop-hdfs-namenode
```

Example output:

```
hadoop-hdfs-namenode.service - Hadoop namenode
   Loaded: loaded (/etc/systemd/system/hadoop-hdfs-namenode.service; enabled; vendor
 preset: disabled)
   Active: active (running) since Wed 2021-08-18 21:38:24 UTC; 2s ago
  Process: 13748 ExecStart=/etc/init.d/hadoop-hdfs-namenode start (code=exited,
 status=0/SUCCESS)
 Main PID: 13800 (java)
    Tasks: 0
   Memory: 1.1M
   CGroup: /system.slice/hadoop-hdfs-namenode.service
           # 13800 /etc/alternatives/jre/bin/java -Dproc_namenode -Xmx1843m -server -
XX:OnOutOfMemoryError=kill -9 %p...
```

EMR 4.x - 5.29.0

Example : Stop a running process

The following example stops the `hadoop-hdfs-namenode` service.

```
sudo stop hadoop-hdfs-namenode
```

Example : Restart a stopped process

The following example restarts the `hadoop-hdfs-namenode` service. You must use the `start` command and not `restart`.

```
sudo start hadoop-hdfs-namenode
```

Example : Check the process status

The following fetches the status for `hadoop-hdfs-namenode`. You can use the `status` command to verify that the process has stopped or started.

```
sudo status hadoop-hdfs-namenode
```

EMR 2.x - 3.x

Example : Stop an application process

The following example stops the `hadoop-hdfs-namenode` service, which is associated with the version of Amazon EMR installed on the cluster.

```
sudo /etc/init.d/hadoop-hdfs-namenode stop
```

Example : Restart an application process

The following example command restarts the `hadoop-hdfs-namenode` process:

```
sudo /etc/init.d/hadoop-hdfs-namenode start
```

Example : Stop an Amazon EMR process

The following example stops a process, such as instance-controller, which is not associated with the version of Amazon EMR on the cluster.

```
sudo /sbin/stop instance-controller
```

Example : Restart an Amazon EMR process

The following example restarts a process, such as instance-controller, which is not associated with the version of Amazon EMR on the cluster.

```
sudo /sbin/start instance-controller
```

> **Note**
> The `/sbin/start, stop` and `restart` commands are symlinks to `/sbin/intictl`. 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. 657) instead.

Topics

- Step 1: Gather data about the issue (p. 654)
- Step 2: Check the environment (p. 654)
- Step 3: Look at the last state change (p. 655)
- Step 4: Examine the log files (p. 655)
- Step 5: Test the cluster step by step (p. 656)

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. 550).

- 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. 654)
- Check usage limits (p. 655)
- Check the release version (p. 655)
- Check the Amazon VPC subnet configuration (p. 655)

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. 306). 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. 550).

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. 564).

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 continue or terminate after step execution (p. 167) and Using termination protection (p. 172).
2. Submit a step to the cluster. For more information, see Submit work to a cluster (p. 642).
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. 655). 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. 172).

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. 653)

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. 195).

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. 657)
- Step 2: Check the environment (p. 658)
- Step 3: Examine the log files (p. 659)
- Step 4: Check cluster and instance health (p. 660)
- Step 5: Check for suspended groups (p. 661)
- Step 6: Review configuration settings (p. 661)
- Step 7: Examine input data (p. 663)

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. 550).

- 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. 658)
- Check usage limits (p. 658)
- Check the Amazon VPC subnet configuration (p. 659)
- Restart the cluster (p. 659)

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. 306). 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. 564).

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. 634).
- **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. 634).

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. 577).

Check job status and HDFS health

Use the **Application user interfaces** tab 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. 557).

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. 599).

- 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. 568).

Step 5: Check for suspended groups

An instance group becomes suspended 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 `SUSPENDED` 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 `SUSPENDED` 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 a suspended state, see Suspended state (p. 637).

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. 191). 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
dfs.replication	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.
io.sort.mb	Total memory available for sorting. This value should be 10x io.sort.factor. This setting can also be used to calculate total memory used by task node by figuring io.sort.mb multiplied by mapred.tasktracker.ap.tasks.maximum.
io.sort.spill.percent	Used during sort, at which point the disk will start to be used because the allotted sort memory is getting full.
mapred.child.java.opts	Deprecated. Use mapred.map.child.java.opts and mapred.reduce.child.java.opts instead. The Java options 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.

Configuration setting	Description
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. 663)
- Permissions errors (p. 665)
- Resource errors (p. 666)
- Streaming cluster errors (p. 673)
- Custom JAR cluster errors (p. 674)
- Hive cluster errors (p. 674)
- VPC errors (p. 675)
- AWS GovCloud (US-West) errors (p. 678)
- Other issues (p. 678)

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. 664)
- Are you trying to recursively traverse input directories? (p. 664)
- Does your output directory already exist? (p. 664)

- Are you trying to specify a resource using an HTTP URL? (p. 664)
- Are you referencing an Amazon S3 bucket using an invalid name format? (p. 664)
- Are you experiencing trouble loading data to or from Amazon S3? (p. 664)

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://*DOC-EXAMPLE-BUCKET1*", you should use "s3n://*DOC-EXAMPLE-BUCKET1*/", 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 "*DOC-EXAMPLE-BUCKET1*.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 User 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. This is also useful when you are troubleshooting a slow cluster. You do this by setting the `mapreduce.map.speculative` and `mapreduce.reduce.speculative` properties to `false`. When you launch a cluster, you can set these values using the `mapred-env` configuration classification. For more information, see Configuring Applications in the *Amazon EMR Release Guide*.
- If you are running a Hive cluster, see Are you having trouble loading data to or from Amazon S3 into Hive? (p. 675).

For additional information, see Amazon S3 error best practices in the *Amazon Simple Storage Service User Guide*.

Permissions errors

The following errors are common when using permissions or credentials.

Topics

- Are you passing the correct credentials into SSH? (p. 665)
- If you are using IAM, do you have the proper Amazon EC2 policies set? (p. 666)

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. 593).

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 How Amazon EMR works with IAM (p. 386).

Resource errors

The following errors are commonly caused by constrained resources on the cluster.

Topics

- Cluster terminates with NO_SLAVE_LEFT and core nodes FAILED_BY_MASTER (p. 666)
- Cannot replicate block, only managed to replicate to zero nodes. (p. 668)
- EC2 QUOTA EXCEEDED (p. 669)
- Too many fetch-failures (p. 669)
- File could only be replicated to 0 nodes instead of 1 (p. 670)
- Deny-listed nodes (p. 671)
- Throttling errors (p. 671)
- Instance type not supported (p. 672)
- EC2 is out of capacity (p. 672)

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 deny lists 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 deny lists 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 deny-listed 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. 341). 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. 610).

Enable termination protection

Enable termination protection. This way, if a core node is deny listed, 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. 172).

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. 577).

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
yarn.nodemanager.disk-health-checker.interval-ms	120000	The frequency (in seconds) that the disk health checker runs.
yarn.nodemanager.disk-health-checker.min-healthy-disks	0.25	The minimum fraction of the number of disks that must be healthy for NodeManager to launch new containers. This corresponds to both yarn.nodemanager.local-dirs (by default, `/mnt/yarn` in Amazon EMR) and yarn.nodemanager.log-dirs (by default `/var/log/hadoop-yarn/containers`, which is symlinked to `mnt/var/log/hadoop-yarn/containers` in Amazon EMR).
`yarn.nodemanager.disk-health-checker.max-disk-utilization-per-disk-percentage`	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 `yarn-nodemanager.local-dirs` and `yarn.nodemanager.log-dirs`.
`yarn.nodemanager.disk-health-checker.min-free-space-per-disk-mb`	0	The minimum space that must be available on a disk for it to be used. This applies to `yarn-nodemanager.local-dirs` and `yarn.nodemanager.log-dirs`.

Cannot replicate block, only managed to replicate to zero nodes.

The error, "Cannot replicate block, only managed to replicate to zero nodes." typically occurs when a cluster does not have enough HDFS storage. This error occurs when you generate more data in your cluster than can be stored in HDFS. You see this error only while the cluster is running, because when the job 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. Task nodes are not used for HDFS storage. All of the disk space on each Amazon EC2 instance, including attached EBS storage volumes, is available to 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.

EC2 QUOTA EXCEEDED

If you get an `EC2 QUOTA EXCEEDED` message, there may be several causes. 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. This message can also be caused by the resizing of an instance group or instance fleet to a target size that is greater than the current instance quota for the account. This can happen manually or automatically through automatic scaling.

Consider the following options to resolve the issue:

- Follow the instructions in AWS service quotas in the *Amazon Web Services General Reference* to request a service limit increase. For some APIs, setting up a CloudWatch event might be a better option than increasing limits. For more details, see When to set up EMR events in CloudWatch (p. 688).
- If one or more running clusters are not at capacity, resize instance groups or reduce target capacities on instance fleets for running clusters.
- Create clusters with fewer EC2 instances or reduced target capacity.

Too many fetch-failures

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. 191).
- 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 number 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. 195).
- 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.

File could only be replicated to 0 nodes instead of 1

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. 634) and Monitor metrics with CloudWatch (p. 577).

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. 564).

Deny-listed nodes

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 deny lists 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 (600,000 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 deny list 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*.

Throttling errors

The errors "Throttled from *`Amazon EC2`* while launching cluster" and "Failed to provision instances due to throttling from *`Amazon EC2`*" occur when Amazon EMR cannot complete a request because another service has throttled the activity. Amazon EC2 is the most common source of throttling errors, but other services may be the cause of throttling errors. AWS service limits apply on a per-Region basis to improve performance, and a throttling error indicates that you have exceeded the service limit for your account in that Region.

Possible causes

The most common source of Amazon EC2 throttling errors is a large number of cluster instances launching so that your service limit for EC2 instances is exceeded. Cluster instances may launch for the following reasons:

- New clusters are created.
- Clusters are resized manually. For more information, see Manually resizing a running cluster (p. 634).
- Instance groups in a cluster add instances (scale out) as a result of an automatic scaling rule. For more information, see Understanding automatic scaling rules (p. 626).
- Instance fleets in a cluster add instances to meet an increased target capacity. For more information, see Configure instance fleets (p. 316).

It is also possible that the frequency or type of API request being made to Amazon EC2 causes throttling errors. For more information about how Amazon EC2 throttles API requests, see Query API request rate in the *Amazon EC2 API Reference*.

Solutions

Consider the following solutions:

- Follow the instructions in AWS service quotas in the *Amazon Web Services General Reference* to request a service limit increase. For some APIs, setting up a CloudWatch event might be a better option than increasing limits. For more details, see When to set up EMR events in CloudWatch (p. 688).
- If you have clusters that launch on the same schedule—for example, at the top of the hour—consider staggering start times.
- If you have clusters that are sized for peak demand, and you periodically have instance capacity, consider specifying automatic scaling to add and remove instances on-demand. In this way, instances are used more efficiently, and depending on the demand profile, fewer instances may be requested at a given time across an account. For more information, see Using automatic scaling with a custom policy for instance groups (p. 625).

Instance type not supported

If you create a cluster, and it fails with the error message "The requested instance type *`InstanceType`* is not supported in the requested Availability Zone," it means that you created the cluster and specified an instance type for one or more instance groups that is not supported by Amazon EMR in the Region and Availability Zone where the cluster was created. Amazon EMR may support an instance type in one Availability Zone within a Region and not another. The subnet you select for a cluster determines the Availability Zone within the Region.

Solution

Determine available instance types in an Availability Zone using the AWS CLI

- Use the `ec2 run-instances` command with the `--dry-run` option. In the example below, replace *`m5.xlarge`* with the instance type you want to use, *`ami-035be7bafff33b6b6`* with the AMI associated with that instance type, and *`subnet-12ab3c45`* with a subnet in the Availability Zone you want to query.

  ```
  aws ec2 run-instances --instance-type m5.xlarge --dry-run --image-
  id ami-035be7bafff33b6b6 --subnet-id subnet-12ab3c45
  ```

  For instructions on finding an AMI ID, see Find a Linux AMI. To find a subnet ID, you can use the describe-subnets command.

To learn more about how to discover available instance types, see Find an Amazon EC2 instance type.

After you determine the instance types available, you can do any of the following:

- Create the cluster in the same Region and EC2 Subnet, and choose a different instance type with similar capabilities as your initial choice. For a list of supported instance types, see Supported instance types (p. 197). To compare capabilities of EC2 instance types, see Amazon EC2 instance types.
- Choose a subnet for the cluster in an Availability Zone where the instance type is available and supported by Amazon EMR.

EC2 is out of capacity

An "EC2 is out of capacity for *`InstanceType`*" error occurs when you attempt to create a cluster, or add instances to a cluster, in an Availability Zone which has no more of the specified EC2 instance type. The subnet that you select for a cluster determines the Availability Zone.

To create a cluster, do one of the following:

- Specify a different instance type with similar capabilities
- Create the cluster in a different Region
- Select a subnet in an Availability Zone where the instance type you want might be available.

To add instances to a running cluster, do one of the following:

- Modify instance group configurations or instance fleet configurations to add available instance types with similar capabilities. For a list of supported instance types, see Supported instance types (p. 197). To compare capabilities of EC2 instance types, see Amazon EC2 instance types.
- Terminate the cluster and recreate it in a Region and Availability Zone where the instance type is available.

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. 673)
- Is your script timing out? (p. 673)
- Are you passing in invalid streaming arguments? (p. 673)
- Did your script exit with an error? (p. 674)

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. 564).

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. 564).

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. 564).

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. 674)
- Is your JAR throwing an error inside a map task? (p. 674)

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. 564).

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. 564).

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. 675)
- Did you encounter a syntax error in the Hive script? (p. 675)
- Did a job fail when running interactively? (p. 675)
- Are you having trouble loading data to or from Amazon S3 into Hive? (p. 675)

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. 564).

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. 564).

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. 664). 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. 676)
- Missing DHCP Options Set (p. 676)
- Permissions errors (p. 676)

- Errors that result in START_FAILED (p. 677)
- Cluster Terminated with errors and NameNode fails to start (p. 677)

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):
PriviledgedActionException 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 service roles for Amazon EMR permissions to AWS services and resources (p. 394).
- 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. 675).

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. 663).

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.
- You are viewing the correct region.

Troubleshoot a Lake Formation cluster

This section walks you through the process of troubleshooting common issues when using Amazon EMR with AWS Lake Formation.

Data lake access not allowed

You must explicitly opt in to data filtering on Amazon EMR clusters before you can analyze and process data in your data lake. When data access fails, you will see a generic `Access is not allowed` message in the output of your notebook entries.

To opt in and allow data filtering on Amazon EMR, see Allow data filtering on Amazon EMR in the *AWS Lake Formation Developer Guide* for instructions.

Session expiration

The session timeout for EMR Notebooks and Zeppelin is controlled by the IAM Role for Lake Formation's `Maximum CLI/API session duration` setting. The default value for this setting is one hour. When a session timeout occurs, you will see the following message in the output of your notebook entries when trying to run Spark SQL commands.

```
Error 401    HTTP ERROR: 401 Problem accessing /sessions/2/statements.
Reason:  JWT token included in request failed validation.
Powered by Jetty:// 9.3.24.v20180605
 org.springframework.web.client.HttpClientErrorException: 401 JWT token included in request
 failed validation…
```

To validate your session, refresh the page. You will be prompted to re-authenticate using your IdP and be redirected back to the Notebook. You can continue to run queries after re-authentication.

No permissions for user on requested table

When attempting to access a table that you do not have access to, you will see the following exception in the output of your notebook entries when trying to run Spark SQL commands.

```
org.apache.spark.sql.AnalysisException: org.apache.hadoop.hive.ql.metadata.HiveException:
 Unable to fetch table table.
Resource does not exist or requester is not authorized to access requested permissions.
(Service: AWSGlue; Status Code: 400; Error Code: AccessDeniedException; Request ID: …
```

To access the table, you must grant access to the user by updating the permissions associated with this table in Lake Formation.

Inserting into, creating and altering tables: Unsupported in beta

Inserting into, creating, or altering tables in databases protected by Lake Formation policies is not supported. When performing these operations, you will see the following exception in the output of your notebook entries when trying to run Spark SQL commands:

```
java.io.IOException:
 com.amazon.ws.emr.hadoop.fs.shaded.com.amazonaws.services.s3.model.AmazonS3Exception:
```

```
            Access Denied (Service: Amazon S3; Status Code: 403; Error Code: AccessDenied;
Request ID: …
```

For more information, see Limitations of Amazon EMR integration with AWS Lake Formation.

Write applications that launch and manage clusters

Topics

- End-to-end Amazon EMR Java source code sample (p. 681)
- Common concepts for API calls (p. 683)
- Use SDKs to call Amazon EMR APIs (p. 685)
- Manage Amazon EMR Service Quotas (p. 687)

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. 685).

> **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. 685).

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 plugin for Eclipse.

To install the Database Development Eclipse plugin

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 (JobFlowId), *`j-xxxxxxxxxxxx`*, 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://path/to/my/jarfolder`*, 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 com.amazonaws.AmazonClientException;
import com.amazonaws.auth.AWSCredentials;
```

```
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduce;
import com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceClientBuilder;
import com.amazonaws.services.elasticmapreduce.model.*;
import com.amazonaws.services.elasticmapreduce.util.StepFactory;

public class Main {

 public static void main(String[] args) {
  AWSCredentials credentials_profile = null;
  try {
   credentials_profile = new ProfileCredentialsProvider("default").getCredentials();
        } catch (Exception e) {
            throw new AmazonClientException(
                    "Cannot load credentials from .aws/credentials file. " +
                    "Make sure that the credentials file exists and the profile name is
 specified within it.",
                    e);
        }

  AmazonElasticMapReduce emr = AmazonElasticMapReduceClientBuilder.standard()
   .withCredentials(new AWSStaticCredentialsProvider(credentials_profile))
   .withRegion(Regions.US_WEST_1)
   .build();

  // Run a bash script using a predefined step in the StepFactory helper class
     StepFactory stepFactory = new StepFactory();
     StepConfig runBashScript = new StepConfig()
       .withName("Run a bash script")
       .withHadoopJarStep(stepFactory.newScriptRunnerStep("s3://jeffgoll/emr-scripts/
create_users.sh"))
       .withActionOnFailure("CONTINUE");

     // Run a custom jar file as a step
     HadoopJarStepConfig hadoopConfig1 = new HadoopJarStepConfig()
        .withJar("s3://path/to/my/jarfolder") // replace with the location of the jar
 to run as a step
        .withMainClass("com.my.Main1") // optional main class, this can be omitted if
 jar above has a manifest
        .withArgs("--verbose"); // optional list of arguments to pass to the jar
     StepConfig myCustomJarStep = new StepConfig("RunHadoopJar", hadoopConfig1);

     AddJobFlowStepsResult result = emr.addJobFlowSteps(new AddJobFlowStepsRequest()
    .withJobFlowId("j-xxxxxxxxxxxx") // replace with cluster id to run the steps
    .withSteps(runBashScript,myCustomJarStep));

          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. 684)
- Specifying cluster parameters in Amazon EMR (p. 684)
- Availability Zones in Amazon EMR (p. 684)
- How to use additional files and libraries in Amazon EMR clusters (p. 685)

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. 685)

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.

```
import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
```

```
import com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduce;
import com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceClientBuilder;
import com.amazonaws.services.elasticmapreduce.model.*;
import com.amazonaws.services.elasticmapreduce.util.StepFactory;

public class Main {

 public static void main(String[] args) {
  AWSCredentialsProvider profile = null;
  try {
   credentials_profile = new ProfileCredentialsProvider("default"); // specifies any named
 profile in .aws/credentials as the credentials provider
        } catch (Exception e) {
            throw new AmazonClientException(
                    "Cannot load credentials from .aws/credentials file. " +
                    "Make sure that the credentials file exists and that the profile name
 is defined within it.",
                    e);
        }

  // create an EMR client using the credentials and region specified in order to create the
 cluster
  AmazonElasticMapReduce emr = AmazonElasticMapReduceClientBuilder.standard()
   .withCredentials(credentials_profile)
   .withRegion(Regions.US_WEST_1)
   .build();

        // create a step to enable debugging in the AWS Management Console
  StepFactory stepFactory = new StepFactory();
  StepConfig enabledebugging = new StepConfig()
      .withName("Enable debugging")
      .withActionOnFailure("TERMINATE_JOB_FLOW")
      .withHadoopJarStep(stepFactory.newEnableDebuggingStep());

        // specify applications to be installed and configured when EMR creates the cluster
  Application hive = new Application().withName("Hive");
  Application spark = new Application().withName("Spark");
  Application ganglia = new Application().withName("Ganglia");
  Application zeppelin = new Application().withName("Zeppelin");

  // create the cluster
  RunJobFlowRequest request = new RunJobFlowRequest()
          .withName("MyClusterCreatedFromJava")
          .withReleaseLabel("emr-5.20.0") // specifies the EMR release version label, we
 recommend the latest release
          .withSteps(enabledebugging)
          .withApplications(hive,spark,ganglia,zeppelin)
          .withLogUri("s3://path/to/my/emr/logs") // a URI in S3 for log files is required
 when debugging is enabled
          .withServiceRole("EMR_DefaultRole") // replace the default with a custom IAM
 service role if one is used
          .withJobFlowRole("EMR_EC2_DefaultRole") // replace the default with a custom EMR
 role for the EC2 instance profile if one is used
          .withInstances(new JobFlowInstancesConfig()
              .withEc2SubnetId("subnet-12ab34c56")
              .withEc2KeyName("myEc2Key")
              .withInstanceCount(3)
              .withKeepJobFlowAliveWhenNoSteps(true)
              .withMasterInstanceType("m4.large")
              .withSlaveInstanceType("m4.large"));

    RunJobFlowResult result = emr.runJobFlow(request);
    System.out.println("The cluster ID is " + result.toString());

 }
```

```
}
```

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 command to create them:

```
aws emr create-default-roles
```

Manage Amazon EMR Service Quotas

Topics

- What are Amazon EMR Service Quotas (p. 687)
- How to manage Amazon EMR Service Quotas (p. 688)
- When to set up EMR events in CloudWatch (p. 688)

The topics in this section describe EMR service quotas (formerly referred to as service limits), how to manage them in the AWS Management Console, and when it's advantageous to use CloudWatch events instead of service quotas to monitor clusters and trigger actions.

What are Amazon EMR Service Quotas

Your AWS account has default service quotas, also known as limits, for each AWS service. The EMR service has two types of limits:

- *Limits on resources* - You can use EMR to create EC2 resources. However, these EC2 resources are subject to service quotas. The resource limitations in this category are:
  - The maximum number of active clusters that can be run at the same time.
  - The maximum number of active instances per instance group.
- *Limits on APIs* - When using EMR APIs, the two types of limitations are:
  - *Burst limit* – This is the maximum number of API calls you can make at once. For example, the maximum number of AddInstanceFleet API requests that you can make per second is set at 5 calls/second as a default. This implies that the burst limit of AddInstanceFleet API is 5 calls/second, or that, at any given time, you can make at most 5 AddInstanceFleet API calls. However, after you use the burst limit, your subsequent calls are limited by the rate limit.
  - *Rate limit* – This is the replenishment rate of the API's burst capacity. For example, replenishment rate of AddInstanceFleet calls is set at 0.5 calls/second as a default. This means that after you reach the burst limit, you have to wait at least 2 seconds (0.5 calls/second X 2 seconds = 1 call) to make the API call. If you make a call before that, you are throttled by the EMR web service. At any point,

you can only make as many calls as the burst capacity without being throttled. Every additional second you wait, your burst capacity increases by 0.5 calls until it reaches the maximum limit of 5, which is the burst limit.

How to manage Amazon EMR Service Quotas

Service Quotas is an AWS feature that you can use to view and manage your Amazon EMR service quotas, or limits, from a central location using the AWS Management Console, the API or the CLI. To learn more about viewing quotas and requesting increases, see AWS service quotas in the *Amazon Web Services General Reference*.

For some APIs, setting up a CloudWatch event might be a better option than increasing service quotas. You can also save time by using CloudWatch to set alarms and trigger increase requests proactively, before you reach the service quota. For more details, see When to set up EMR events in CloudWatch (p. 688).

When to set up EMR events in CloudWatch

For some polling APIs, such as DescribeCluster, DescribeStep, and ListClusters, setting up a CouldWatch event can reduce the response time to changes and free up your service quotas. For example, if you have a Lambda function set up to run when a cluster's state changes, such as when a step completes or a cluster terminates, you can use that trigger to start the next action in your workflow instead of waiting for the next poll. Otherwise, if you have dedicated Amazon EC2 instances or Lambda functions constantly polling the EMR API for changes, you not only waste compute resources but might also reach your service quota.

Following are a few cases when you might benefit by moving to an event-driven architecture.

Case 1: Polling EMR using DescribeCluster API calls for step completion

Example Polling EMR using DescribeCluster API calls for step completion

A common pattern is to submit a step to a running cluster and poll Amazon EMR for status about the step, typically using the DescribeCluster or DescribeStep APIs. This task can also be accomplished with minimal delay by hooking into Amazon EMR Step Status Change event in.

This event includes the following information in its payload.

```
{
  "version": "0",
  "id": "999cccaa-eaaa-0000-1111-123456789012",
  "detail-type": "EMR Step Status Change",
  "source": "aws.emr",
  "account": "123456789012",
  "time": "2016-12-16T20:53:09Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "severity": "ERROR",
    "actionOnFailure": "CONTINUE",
    "stepId": "s-ZYXWVUTSRQPON",
    "name": "CustomJAR",
    "clusterId": "j-123456789ABCD",
    "state": "FAILED",
    "message": "Step s-ZYXWVUTSRQPON (CustomJAR) in Amazon EMR cluster j-123456789ABCD
 (Development Cluster) failed at 2016-12-16 20:53 UTC."
```

```
  }
}
```

In the detail map, a Lambda function could parse for "state", "stepId", or "clusterId" to find pertinent information.

Case 2: Polling EMR for available clusters to run workflows

Example Polling EMR for available clusters to run workflows

A pattern for customers who run multiple clusters is to run workflows on clusters as soon as they're available. If there are many clusters running and a workflow needs to be performed on a cluster that's waiting, a pattern could be to poll EMR using DescribeCluster or ListClusters API calls for available clusters. Another way to reduce the delay in knowing when a cluster is ready for a step, would be to process Amazon EMR Cluster State Change event in.

This event includes the following information in its payload.

```
{
  "version": "0",
  "id": "999cccaa-eaaa-0000-1111-123456789012",
  "detail-type": "EMR Cluster State Change",
  "source": "aws.emr",
  "account": "123456789012",
  "time": "2016-12-16T20:43:05Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "severity": "INFO",
    "stateChangeReason": "{\"code\":\"\"}",
    "name": "Development Cluster",
    "clusterId": "j-123456789ABCD",
    "state": "WAITING",
    "message": "Amazon EMR cluster j-123456789ABCD ..."
  }
}
```

For this event, a Lambda function could be set up to immediately send a waiting workflow to a cluster as soon as its status changes to WAITING.

Case 3: Polling EMR for cluster termination

Example Polling EMR for cluster termination

A common pattern of customers running many EMR clusters is polling Amazon EMR for terminated clusters so that work is no longer sent to it. You can implement this pattern with the DescribeCluster and ListClusters API calls or by using Amazon EMR Cluster State Change event in.

Upon cluster termination, the event emitted looks like the following example.

```
{
  "version": "0",
  "id": "1234abb0-f87e-1234-b7b6-000000123456",
  "detail-type": "EMR Cluster State Change",
  "source": "aws.emr",
  "account": "123456789012",
  "time": "2016-12-16T21:00:23Z",
  "region": "us-east-1",
  "resources": [],
```

```
  "detail": {
    "severity": "INFO",
    "stateChangeReason": "{\"code\":\"USER_REQUEST\",\"message\":\"Terminated by user
 request\"}",
    "name": "Development Cluster",
    "clusterId": "j-123456789ABCD",
    "state": "TERMINATED",
    "message": "Amazon EMR Cluster jj-123456789ABCD (Development Cluster) has terminated at
 2016-12-16 21:00 UTC with a reason of USER_REQUEST."
  }
}
```

The "detail" section of the payload includes the clusterId and state that can be acted on.

AWS glossary

For the latest AWS terminology, see the AWS glossary in the *AWS General Reference*.