Management Guide

Amazon EMR

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon EMR: Management Guide

Copyright © 2025 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
Understanding how to create and work with Amazon EMR clusters 1
Getting familiar with clusters and nodes 2
Submitting work to a cluster 2
Processing data 3
Understanding the cluster lifecycle 4
Benefits of using Amazon EMR 6
Cost savings 7
AWS integration 7
Deployment 8
Scalability and flexibility 8
Reliability 9
Security 9
Monitoring 11
Management interfaces 11
Architecture and service layers 12
Storage 13
Cluster resource management 13
Data processing frameworks 14
Applications and programs 15
Before you set up Amazon EMR 16
Sign up for an AWS account 16
Create a user with administrative access 16
Create an Amazon EC2 key pair for SSH 18
Next steps 18
Getting started tutorial 19
Setting up your Amazon EMR cluster 19
Step 1: Configure data resources and launch an Amazon EMR cluster 20
Prepare storage for Amazon EMR 20
Prepare an application with input data for Amazon EMR 21
Launch an Amazon EMR cluster 23
Step 2: Submit work to your Amazon EMR cluster 26
Submit work and view results 26
View results 29

Step 3: Clean up ... 33
Terminate your cluster ... 33
Delete S3 resources ... 35
Next steps ... 36
Explore big data applications for Amazon EMR ... 36
Plan cluster hardware, networking, and security ... 36
Manage clusters ... 36
Use a different interface ... 36
Browse the EMR technical blog ... 37
Managing Amazon EMR clusters with the console ... 38
Console capabilities ... 38
Summary of differences ... 39
Cluster compatibility in the console ... 39
Creating clusters ... 39
Viewing and searching for clusters ... 41
Viewing or editing cluster details ... 42
Differences when you work with security configurations ... 43
Amazon EMR Studio ... 44
Key features ... 44
Amazon SageMaker Unified Studio IDE ... 45
Feature history ... 45
How it works ... 46
Authentication and user login ... 47
Access control ... 50
Workspaces ... 51
Notebook storage ... 52
EMR Studio features, requirements, and limits ... 52
Considerations ... 52
Known issues ... 55
Feature limitations ... 56
Service limits ... 57
VPC and subnet best practices for EMR Studio ... 57
Cluster requirements ... 58
Configure EMR Studio ... 60
Administrator permissions to create an EMR Studio ... 61
Set up an EMR Studio ... 67

Monitor, update and delete Amazon EMR Studio resources ... 137
Encrypting workspace notebooks ... 144
Control EMR Studio network traffic ... 146
Create cluster templates ... 148
Access and permissions for Git-based repositories ... 154
Optimize Spark jobs ... 157
Use an EMR Studio ... 159
Learn EMR Studio workspaces ... 159
Workspace collaboration ... 166
Run Workspace with a runtime role ... 169
Run Amazon EMR Studio Workspace Workspace notebooks programmatically ... 175
Browse data with SQL Explorer for EMR Studio ... 175
Attach a compute to a Workspace ... 176
Link Git repositories ... 183
Athena integration ... 187
CodeWhisperer integration ... 188
Debug applications and jobs ... 190
Install kernels and libraries ... 194
Magic commands ... 195
Use multi-language notebooks with Spark kernels ... 204
EMR Notebooks ... 207
Notebooks in the console ... 208
About the transition ... 208
What do you need to do? ... 208
Workspace advantages ... 209
Required permissions ... 209
Considerations ... 211
Cluster requirements ... 211
Differences in capabilities by cluster release version ... 212
Limits for concurrently attached EMR Notebooks ... 213
Jupyter Notebook and Python versions ... 214
Security-related considerations ... 214
Create a Notebook ... 214
Working with EMR Notebooks ... 217
Understanding Notebook status ... 218
Working with the Notebook editor ... 219

Changing clusters ... 220
Deleting Notebooks and Notebook files ... 221
Sharing Notebook files ... 222
Sample programmatic commands for EMR Notebooks ... 223
Overview ... 223
Permissions ... 223
Limitations ... 225
Examples ... 225
Notebook CLI command samples ... 226
Boto3 SDK sample script ... 232
Ruby sample script ... 235
User impersonation for Spark ... 237
Setting up Spark user impersonation ... 237
Using the Spark job monitoring widget ... 238
Security ... 239
Installing and using kernels and libraries in EMR Studio ... 240
... 240
Installing kernels and Python libraries on a cluster primary node ... 241
Considerations and limitations with notebook-scoped libraries ... 243
Working with Notebook-scoped libraries ... 244
Associating Git-based repositories with EMR Notebooks ... 245
Prerequisites and considerations ... 246
Add a Git-based repository to Amazon EMR ... 250
Update or delete a Git-based repository from an EMR Studio Workspace ... 250
Link or unlink a Git-based repository in EMR Studio ... 251
Create a new Notebook with an associated Git repository in EMR Studio ... 252
Use Git repositories in an EMR Studio Notebook ... 253
Plan, configure and launch Amazon EMR clusters ... 254
Launch an Amazon EMR cluster quickly ... 254
Configure Amazon EMR cluster location and data storage ... 255
Choose an AWS Region for your Amazon EMR cluster ... 255
Working with storage and file systems ... 257
Prepare input data for processing with Amazon EMR ... 261
Configure a location for Amazon EMR cluster output ... 280
Plan and configure primary nodes in your Amazon EMR cluster ... 286

Features that support high availability in an Amazon EMR cluster and how they work with open-source applications ... 287
Launch an Amazon EMR Cluster with multiple primary nodes ... 296
Amazon EMR integration with EC2 placement groups ... 302
Considerations and best practices for a cluster with multiple primary nodes ... 310
EMR clusters on AWS Outposts ... 312
Prerequisites ... 312
Limitations ... 313
Network connectivity considerations ... 314
Creating an Amazon EMR cluster on AWS Outposts ... 314
EMR clusters on AWS Local Zones ... 315
Supported instance types ... 316
Creating an Amazon EMR cluster on Local Zones ... 316
Configure Docker for use with Amazon EMR clusters ... 317
Docker registries ... 318
Configuring Docker registries ... 319
Configuring YARN to access Amazon ECR on EMR 6.0.0 and earlier ... 320
Control Amazon EMR cluster termination ... 322
Configuring an Amazon EMR cluster to continue or terminate after step execution ... 323
Using an auto-termination policy ... 325
Using termination protection to protect your Amazon EMR clusters from accidental shut down ... 331
Replacing unhealthy nodes with Amazon EMR ... 337
Default node replacement and termination protection settings ... 339
Configuring unhealthy node replacement when you launch a cluster ... 339
Configuring unhealthy node replacement in a running cluster ... 340
Working with AMIs ... 342
Overview ... 342
Using the default AMI ... 342
Using a custom AMI to provide more flexibility for Amazon EMR cluster configuration ... 520
Change the AL release ... 532
Customizing the EBS root volume ... 533
Configure applications when you launch your Amazon EMR cluster ... 537
Create bootstrap actions ... 537
Configure Amazon EMR cluster hardware and networking ... 542
Understand node types ... 543

Configure Amazon EC2 instance types for use with Amazon EMR 546
Configure Amazon EMR cluster logging and debugging 1526
Default log files 1527
Archive log files to Amazon S3 1528
Log locations 1533
Tag and categorize Amazon EMR cluster resources 1535
Restrictions that apply to tagging resources in Amazon EMR 1536
Tag Amazon EMR resources for billing 1537
Add tags to an Amazon EMR cluster 1537
View tags on an Amazon EMR cluster 1539
Remove tags from an Amazon EMR cluster 1540
Drivers and third-party application integration on Amazon EMR 1541
Use business intelligence tools with Amazon EMR 1541
Security 1543
Network and infrastructure security 1543
Default Amazon Linux AMI updates 1544
AWS Identity and Access Management with Amazon EMR 1544
Single-tenant and multi-tenant clusters 1546
Data protection 1546
Data Access Control 1547
Security configurations 1547
Create a security configuration with the Amazon EMR console or with the AWS CLI 1548
Specify a security configuration 1576
Data protection 1577
Encrypt data at rest and in transit with Amazon EMR 1578
IAM with Amazon EMR 1613
Audience 1614
Authenticating with identities 1614
Managing access using policies 1618
How Amazon EMR works with IAM 1620
Runtime roles for Amazon EMR steps 1627
Configure service roles for Amazon EMR 1636
Identity-based policy examples 1698
S3 Access Grants with Amazon EMR 1740
Overview 1740
How it works 1741

Considerations ... 1741
Launch a cluster ... 1742
`Lake Formation` ... 1744
`fallbackToIAM` ... 1744
Authenticate to cluster nodes ... 1745
Use an EC2 key pair for SSH credentials for Amazon EMR ... 1745
Use Kerberos authentication ... 1746
Use LDAP authentication ... 1783
Integrate Amazon EMR with Identity Center ... 1794
Overview ... 1794
Features ... 1794
Getting started ... 1795
Considerations ... 1803
Integrate Amazon EMR with Lake Formation ... 1805
How Amazon EMR works with Lake Formation ... 1805
Prerequisites ... 1806
Enable Lake Formation with Amazon EMR ... 1807
Hudi and Lake Formation ... 1813
Iceberg and Lake Formation ... 1815
Delta Lake and Lake Formation ... 1817
Considerations ... 1818
Working with Glue Data Catalog views in Amazon EMR ... 1821
Using a Lake Formation job runtime role with full table access ... 1827
Integrate Amazon EMR with Apache Ranger ... 1835
Ranger overview ... 1835
Considerations for using Amazon EMR with Apache Ranger ... 1838
Set up Amazon EMR for Apache Ranger ... 1840
Apache Ranger plugins for Amazon EMR integration scenarios ... 1859
Apache Ranger troubleshooting ... 1885
Control network traffic with security groups for your Amazon EMR cluster ... 1889
Working with Amazon EMR-managed security groups ... 1891
Working with additional security groups for an Amazon EMR cluster ... 1900
Specifying security groups ... 1901
Security groups for EMR Notebooks ... 1904
Block public access ... 1906
Compliance validation ... 1910

Resilience ... 1911
Infrastructure security ... 1911
Connect to Amazon EMR using an interface VPC endpoint ... 1912
Manage Amazon EMR clusters ... 1917
Connect to an Amazon EMR cluster ... 1917
Before you connect ... 1918
Connect to the Amazon EMR cluster primary node using SSH ... 1920
Submit work to an Amazon EMR cluster ... 1944
Add steps with the console ... 1945
Add steps with the CLI ... 1948
Running multiple steps ... 1950
Viewing steps after submitting work to an Amazon EMR cluster ... 1951
Cancel steps when you submit work to an Amazon EMR cluster ... 1952
View and monitor an Amazon EMR cluster as it performs work ... 1953
View Amazon EMR cluster status and details ... 1954
Enhanced step debugging with Amazon EMR ... 1959
View Amazon EMR application history ... 1961
View Amazon EMR log files ... 1973
View cluster instances in Amazon EC2 ... 1977
CloudWatch events and metrics from Amazon EMR ... 1978
View cluster application metrics using Ganglia with Amazon EMR ... 2063
CloudTrail logs ... 2063
EMR Observability Best Practices ... 2068
Use Amazon EMR cluster scaling to adjust for changing workloads ... 2068
Considerations ... 2070
Managed scaling ... 2070
Automatic scaling with a custom policy ... 2112
Resize a running cluster ... 2124
Provisioning timeouts ... 2132
Cluster scale-down options for Amazon EMR clusters ... 2136
Terminate an Amazon EMR cluster in the starting, running, or waiting states ... 2140
Terminate from console ... 2141
Terminate from CLI ... 2141
Terminate from API ... 2143
Clone a cluster ... 2143
Automate recurring Amazon EMR clusters with AWS Data Pipeline ... 2144

Amazon EMR tutorials 2145
Amazon EMR on EC2 – Enhanced Monitoring with CloudWatch using custom metrics and logs 2145
Overview 2145
Prerequisites and Background 2145
..... 2146
Monitor Apache Spark applications on Amazon EMR with Amazon CloudWatch 2158
Monitor Amazon EMR application status with CloudWatch integration 2159
Troubleshoot Amazon EMR clusters 2160
Troubleshooting tools 2160
View cluster details 2161
View error details 2161
Run scripts and configure processes 2162
View log files 2162
Monitor cluster performance 2163
View and restart processes 2163
Viewing running processes 2163
Stopping and restarting processes 2165
Common errors 2168
Error codes 2169
Resource errors during Amazon EMR cluster operations 2186
Cluster input and output errors during Amazon EMR operations 2199
Permissions errors during Amazon EMR cluster operations 2202
Hive cluster errors 2203
VPC errors during Amazon EMR cluster operations 2205
Streaming Amazon EMR cluster errors 2209
Amazon EMR: Custom JAR cluster errors 2210
Amazon EMR AWS GovCloud (US-West) errors 2211
Find a missing cluster 2211
Troubleshoot failed clusters 2212
Step 1: Gather data about the issue with the Amazon EMR cluster 2212
Step 2: Check the environment 2213
Step 3: Look at the last state change 2214
Step 4: Examine the Amazon EMR log files 2215
Step 5: Test the Amazon EMR cluster step by step 2216
Troubleshoot a slow cluster 2217

Step 1: Gather data about the issue with the Amazon EMR cluster 2218
Step 2: Check the EMR cluster environment 2218
Step 3: Examine the log files for the Amazon EMR cluster 2220
Step 4: Check Amazon EMR cluster and instance health 2221
Step 5: Check for suspended groups 2223
Step 6: Review configuration settings for the Amazon EMR cluster 2223
Step 7: Examine input data for the Amazon EMR cluster 2226
Troubleshoot common issues when using Amazon EMR with AWS Lake Formation 2226
Data lake access not allowed 2226
Session expiration 2227
No permissions for user on requested table 2227
Querying cross-account data shared with Lake Formation 2227
Inserting into, creating, and altering tables 2229
Write applications that launch and manage Amazon EMR clusters 2230
End-to-end Amazon EMR Java source code sample 2230
Common concepts for Amazon EMR API calls 2234
Endpoints for Amazon EMR 2235
Specifying cluster parameters in Amazon EMR 2235
Availability Zones in Amazon EMR 2235
How to use additional files and libraries in Amazon EMR clusters 2236
Use SDKs to call Amazon EMR APIs 2236
Using the AWS SDK for Java to create an Amazon EMR cluster 2237
Manage Amazon EMR Service Quotas 2239
What are Amazon EMR Service Quotas 2239
How to manage Amazon EMR Service Quotas 2240
When to set up EMR events in CloudWatch 2241
AWS Glossary 2245
Document history 2246

What is Amazon EMR?

Amazon EMR, which was 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 – This tutorial gets you started using Amazon EMR quickly.

In This Section

- Understanding how to create and work with Amazon EMR clusters
- Benefits of using Amazon EMR
- Amazon EMR architecture and service layers

Understanding how to create and work with Amazon EMR clusters

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

- Getting familiar with clusters and nodes
- Submitting work to a cluster
- Processing data
- Understanding the cluster lifecycle

Getting familiar with 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:

- **Primary 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 primary node tracks the status of tasks and monitors the health of the cluster. Every cluster has a primary node, and it's possible to create a single-node cluster with only the primary 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.

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 an Amazon EMR cluster.
- Create a cluster, connect to the primary 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 primary 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 an Amazon EMR cluster.

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.

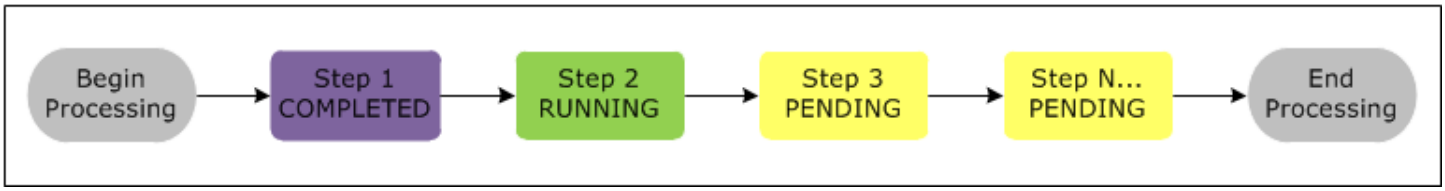
Generally, when you process data in Amazon EMR, the input is data stored as files in your chosen underlying file system, such as Amazon S3 or HDFS. This data passes from one step to the next in the processing sequence. The final step writes the output data to a specified location, such as an Amazon S3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

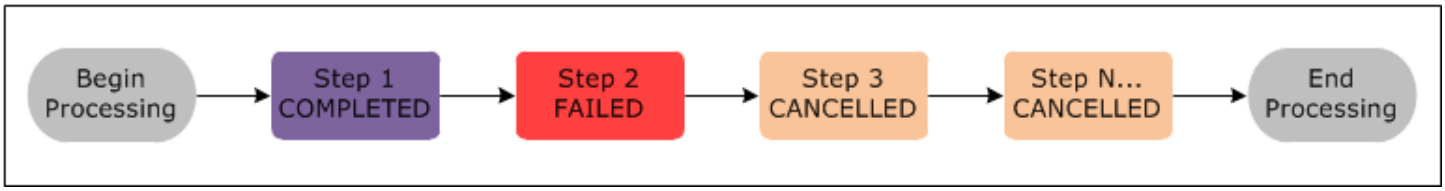
5. The next step in the sequence starts, and its state changes to **RUNNING**. When it completes, its state changes to **COMPLETED**.
6. This pattern repeats for each step until they all complete and processing ends.

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

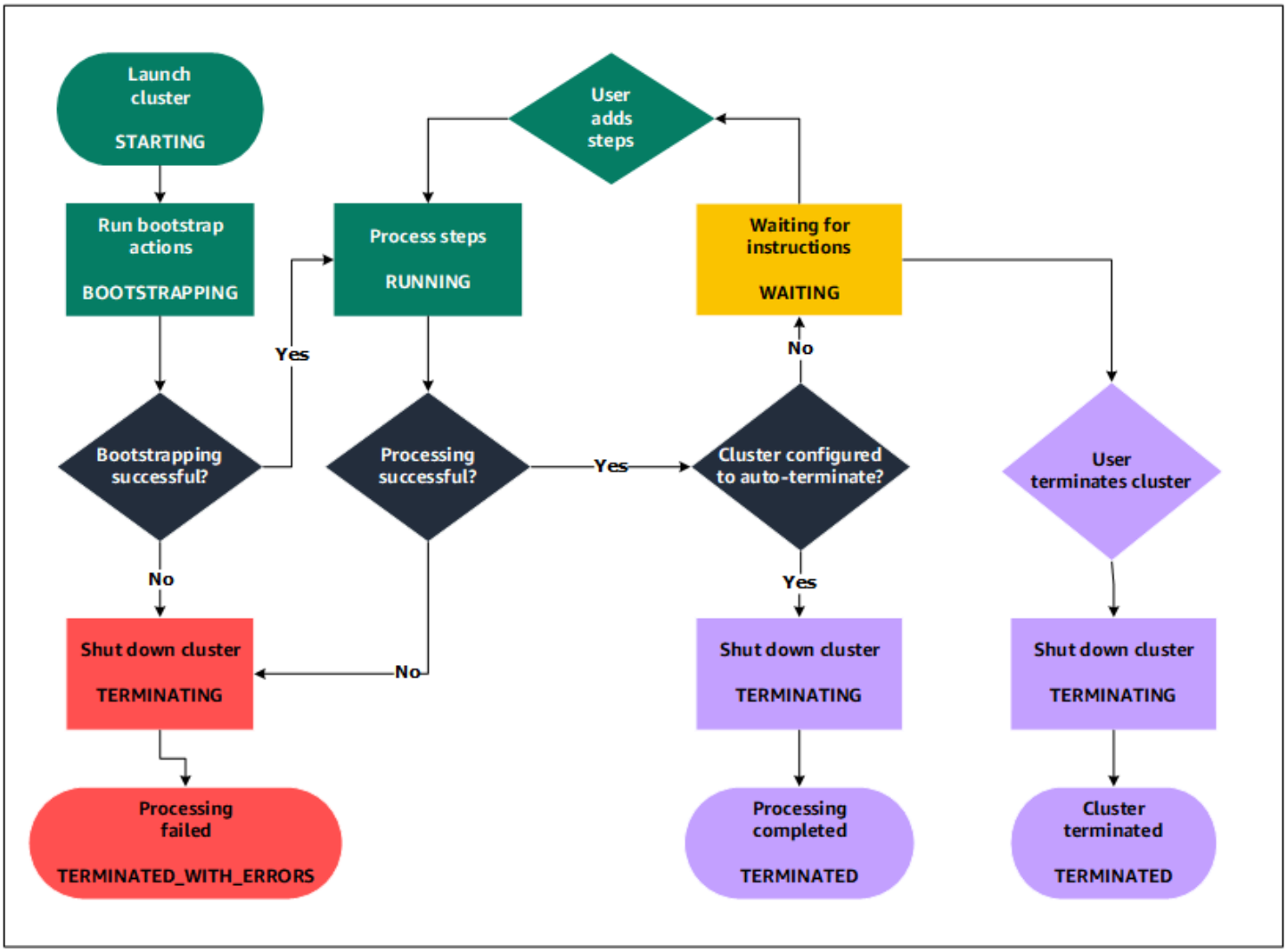
A successful Amazon EMR cluster follows this process:

1. Amazon EMR first provisions EC2 instances in the cluster for each instance according to your specifications. For more information, see Configure Amazon EMR cluster hardware and networking. 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 to provide more flexibility for Amazon EMR cluster configuration. 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 with an Amazon EMR cluster. 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 an Amazon EMR cluster.
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 to protect your Amazon EMR clusters from accidental shut down.

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. These include the flexibility offered through AWS and the cost savings available versus building your own on-premises resources. This section provides an overview of these benefits and links to additional information to help you explore further.

Topics

- Cost savings
- AWS integration
- Deployment
- Scalability and flexibility

- Reliability
- Security
- Monitoring
- Management interfaces

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 Amazon EMR cluster hardware and networking.

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 applications when you launch your Amazon EMR cluster.

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 resize a running Amazon EMR cluster.

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?.

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 primary 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 Working with storage and file systems with Amazon EMR.

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 to protect your Amazon EMR clusters from accidental shut down.

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 a 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.

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.

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 primary 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 primary and core/task instances for more advanced rules. For more information, see Control network traffic with security groups for your Amazon EMR cluster.

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 in a VPC for Amazon EMR.

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 AWS EMR API calls using AWS CloudTrail.

Amazon EC2 key pairs

You can monitor and interact with your cluster by forming a secure connection between your remote computer and the primary 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 EC2 key pair for SSH credentials for Amazon EMR.

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 Amazon EMR cluster logging and debugging.

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 Monitoring Amazon EMR metrics with CloudWatch.

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.

Amazon EMR architecture and service layers

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
- Cluster resource management
- Data processing frameworks
- Applications and programs

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 options and behavior in Amazon EMR 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.

Before you set up Amazon EMR

Complete the preliminary tasks detailed in this section before you launch an Amazon EMR cluster for the first time. These include setting up your AWS account if you need one and and taking steps to set up secure communication.

Sign up for an AWS account

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

To sign up for an AWS account

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

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

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

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

Create a user with administrative access

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

Secure your AWS account root user

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

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

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

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

Create a user with administrative access

1. Enable IAM Identity Center.

   For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.

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

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

Sign in as the user with administrative access

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

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

Assign access to additional users

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

   For instructions, see Create a permission set in the *AWS IAM Identity Center User Guide*.

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

   For instructions, see Add groups in the *AWS IAM Identity Center User Guide*.

Create 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 for authentication with Amazon EMR.

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

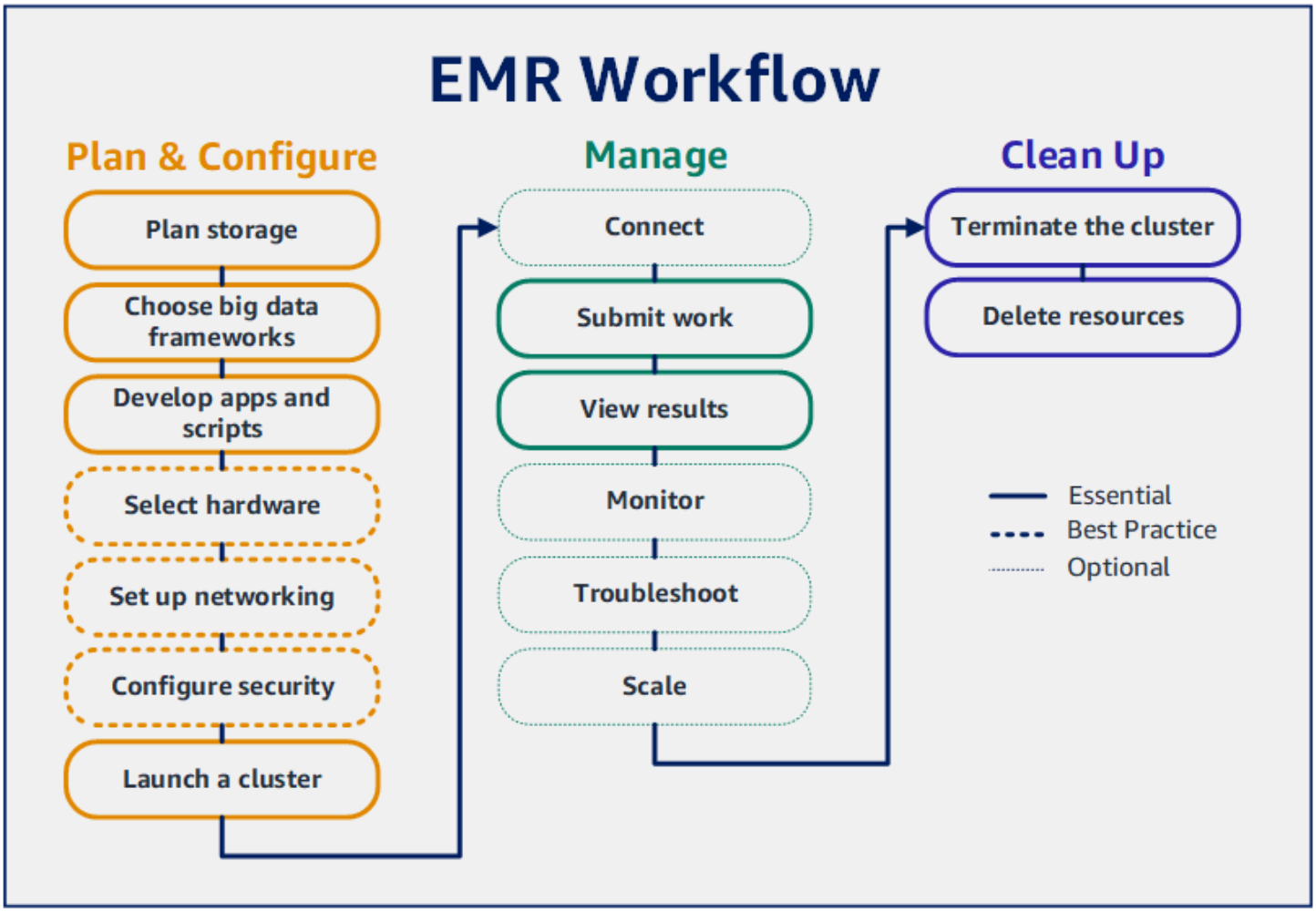
Tutorial: Getting started with Amazon EMR

Walk through a workflow to quickly set up an Amazon EMR cluster and run a Spark application.

Setting up your Amazon EMR cluster

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 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 Before you set up Amazon EMR.

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: Configure data resources and launch 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 Working with storage and file systems with Amazon EMR.

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. Don't download the restaurant data for this tutorial directly from the King County website, because it's a really big file. We provide a download below that has a file with fewer records, in order to help complete the tutorial. 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://
amzn-s3-demo-bucket/food-establishment-data.csv'.
    :param output_uri: The URI where output is written, such as 's3://amzn-s3-demo-
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 for processing with Amazon EMR.

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 with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose **Create cluster**.
3. On the **Create Cluster** page, note the default values for **Release**, **Instance type**, **Number of instances**, and **Permissions**. These fields automatically populate with values that work for general-purpose clusters.
4. In the **Cluster name** field, enter a unique cluster name to help you identify your cluster, such as *`My first cluster`*. Your cluster name can't contain the characters <, >, $, |, or ` (backtick).
5. Under **Applications**, choose the **Spark** option 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.

6. Under **Cluster logs**, select the Publish cluster-specific logs to Amazon S3 check box. Replace the **Amazon S3 location** value with the Amazon S3 bucket you created, followed by **`/logs`**. For example, **`s3://amzn-s3-demo-bucket/logs`**. Adding **`/logs`** creates a new folder called 'logs' in your bucket, where Amazon EMR can copy the log files of your cluster.
7. Under **Security configuration and permissions**, choose your **EC2 key pair**. In the same section, select the **Service role for Amazon EMR** dropdown menu and choose **EMR_DefaultRole**. Then, select the **IAM role for instance profile** dropdown menu and choose **EMR_EC2_DefaultRole**.
8. Choose **Create cluster** to launch the cluster and open the cluster details 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 Amazon EMR cluster status and details. For information about cluster status, see Understanding the cluster lifecycle.

CLI

To launch a cluster with Spark installed with 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-5.36.2> \
--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.

Step 2: Submit work to your Amazon EMR cluster

Submit work and view results

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 an Amazon EMR cluster.

Console

To submit a Spark application as a step with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then select the cluster where you want to submit work. The cluster state must be **Waiting**.
3. Choose the **Steps** tab, and then choose **Add step**.
4. Configure the step according to the following guidelines:
   - For **Type**, choose **Spark application**. You should see additional fields for **Deploy mode**, **Application location**, and **Spark-submit options**.
   - For **Name**, enter 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 mode**. For more information on Spark deployment modes, see Cluster mode overview in the Apache Spark documentation.
- For **Application location**, enter the location of your `health_violations.py` script in Amazon S3, such as *`s3://amzn-s3-demo-bucket/health_violations.py`*.
- Leave the **Spark-submit options** field empty. For more information on `spark-submit` options, see Launching applications with spark-submit.
- In the **Arguments** field, enter the following arguments and values:

```
--data_source s3://amzn-s3-demo-bucket/food_establishment_data.csv
--output_uri s3://amzn-s3-demo-bucket/myOutputFolder
```

  Replace *`s3://amzn-s3-demo-bucket/food_establishment_data.csv`* with the S3 bucket URI of the input data you prepared in Prepare an application with input data for Amazon EMR.

  Replace *`amzn-s3-demo-bucket`* with the name of the bucket that you created for this tutorial, and replace *`myOutputFolder`* with a name for your cluster output folder.
- For **Action if step fails**, accept the default option **Continue**. This way, 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. Monitor the step status. It should change from **Pending** to **Running** to **Completed**. To refresh the status in the console, choose the refresh icon to the right of **Filter**. The script takes about one minute to run. When the status changes to **Completed**, the step has completed successfully.

CLI

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

1. Make sure you have the `ClusterId` of the cluster you launched in Launch an Amazon EMR cluster. 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://amzn-s3-demo-bucket/health_violations.py`* with the location of your `health_violations.py` application.
   - Replace *`s3://amzn-s3-demo-bucket/food_establishment_data.csv`* with the S3 location of your `food_establishment_data.csv` dataset.
   - Replace *`s3://amzn-s3-demo-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://amzn-s3-demo-
bucket/health_violations.py>,--data_source,<s3://amzn-s3-demo-bucket/
food_establishment_data.csv>,--output_uri,<s3://amzn-s3-demo-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://amzn-s3-demo-bucket/health_violations.py",
                "--data_source",
                "s3://amzn-s3-demo-bucket/food_establishment_data.csv",
                "--output_uri",
                "s3://amzn-s3-demo-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.

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, *amzn-s3-demo-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 a location for Amazon EMR cluster output.

(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.

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 primary node. In an Amazon EMR cluster, the primary node is an Amazon EC2 instance that manages the cluster.

ElasticMapReduce-slave

The default security group associated with core and task nodes.

Console

To allow SSH access for trusted sources for the primary security group with the 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 a user and the Example Policy that allows managing EC2 security groups in the *IAM User Guide*.

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose the cluster that you want to update. This opens up the cluster details page. The **Properties** tab on this page should be pre-selected.
3. Under **Networking** in the **Properties** tab, select the arrow next to **EC2 security groups (firewall)** to expand this section. Under **Primary node**, select the security group link. When you've completed the following steps, you can optionally come back to this step, choose **Core and task nodes**, and repeat the following steps to allow SSH client access to core and task nodes.
4. This opens the EC2 console. Choose the **Inbound rules** tab and then **Edit inbound rules**.
5. 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.

6. Scroll to the bottom of the list of rules and choose **Add Rule**.
7. For **Type**, select **SSH**. Selecting SSH automatically enters **TCP** for **Protocol** and **22** for **Port Range**.
8. 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.
9. Choose **Save**.
10. Optionally, choose **Core and task nodes** 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 primary node.

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

Use Amazon SageMaker AI Unified Studio to manage your Amazon EMR cluster

Amazon EMR on EC2 is also a supported compute type for Amazon SageMaker AI Unified Studio. Refer to Managing Amazon EMR on EC2 for how to use and manage EMR on EC2 resources in Amazon SageMaker AI Unified Studio.

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 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 with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Choose **Clusters**, and then choose the cluster you want to terminate.
3. Under the **Actions** dropdown menu, choose **Terminate cluster**.
4. Choose **Terminate** in the dialog box. Depending on the cluster configuration, termination may take 5 to 10 minutes. For more information on how to Amazon EMR clusters, see Terminate an Amazon EMR cluster in the starting, running, or waiting states.

CLI

To terminate the cluster with 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 an Amazon EMR cluster in the starting, running, or waiting states.

```
{
    "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, configure and launch Amazon EMR clusters and Security in Amazon EMR.

Manage clusters

Dive deeper into working with running clusters in Manage Amazon EMR clusters. 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.

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.

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.

Browse the EMR technical blog

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

Managing Amazon EMR clusters with the console

The console offers an updated interface that provides you with an intuitive way to manage your Amazon EMR environment and gives you convenient access to documentation, product information, and other resources.

Console capabilities

The Amazon EMR console is available at the following URL:

- **Console URL** – https://console.aws.amazon.com/emr

The following table lists the main Amazon EMR console components status.

Amazon EMR console component	Console
EMR Studio	✓
Create and manage clusters	✓
Block public access	✓
Monitor Amazon CloudWatch Events	✓
Security configurations	✓
Virtual clusters (Amazon EMR on EKS)	✓
View and manage your Amazon Virtual Private Cloud subnets[1]	✓
Notebooks[2]	✓

[1] In the console, you can view and manage your Amazon VPC subnets within the **Networking** section when you create a cluster.

[2] EMR Notebooks are available as EMR Studio Workspaces in the console. The **Create Workspace** button in the console lets you create new notebooks. To access or create Workspaces, EMR Notebooks users need additional IAM role permissions. For more information, see Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console and Amazon EMR console.

Summary of differences

This section outlines the capabilities of the Amazon EMR console experience. These capabilities fall into the following categories:

- Cluster compatibility in the console
- Creating clusters
- Viewing or editing cluster details
- Viewing and searching for clusters
- Differences when you work with security configurations

Cluster compatibility in the console

In some cases, a cluster that you created might not be compatible with the console. The following list describes compatibility requirements for the Amazon EMR console.

- The console supports clusters created in Amazon EMR releases 5.20.1 and later.
- You can clone clusters that use automatic scaling in the console, but you can only create new clusters if you want to manually scale them or use managed scaling.

To create and work with clusters of release 5.20.1 and earlier, you can use the AWS Command Line Interface (AWS CLI) or the AWS SDK.

Creating clusters

Capability	Console	
	Primary, core, task	

Capability	Console
Terminology: Amazon EMR cluster node types	
Amazon EMR supported releases[1]	Amazon EMR release 5.20.1 and later
Quickly launching a cluster	Use the **Create cluster** button under the **Summary** panel. Your cluster name can't contain the characters <, >, $, |, or ` (backtick).
Configuring a Spot provisioning timeout	Define a timeout period for provisioning instances for each fleet in your cluster.
Service roles and Amazon EC2 instance profile role	The console doesn't create default roles; you must create roles with the IAM Console or select an already-created IAM role
Cluster visibility	From within the Amazon EMR console, you can't make a cluster visible to all a users; your IAM policy determines cluster access
Networking - configure private subnets	You must configure Amazon S3 endpoints and NAT gateways from their respective Amazon S3 and Amazon VPC consoles

Capability	Console	
EMR File System consistent view (EMRFS CV)	With the release of Amazon S3 strong read-after-write consistency on December 1, 2020, you don't need to use EMRFS CV with your EMR clusters	
Debugging	You can debug jobs using the Application UI interface on the cluster details page	

[1] You can't create or edit clusters using releases earlier than Amazon EMR 5.20.1 in the console, but any existing clusters created using releases earlier than 5.20.1 will continue to work. To create and edit clusters with Amazon EMR releases earlier than 5.20.1, use the API or CLI. You can view all clusters using the console, but consoles created earlier than 5.20.1 might not be compatible newer features.

Viewing and searching for clusters

The following table highlights how you can use the Amazon EMR console to view view and search for clusters.

> **Note**
>
> Applying a data filter to the cluster list queries the entire database. But when you enter a text string into the search box, the search only applies to the results that the list has loaded client side.

Capability	Console	
Viewing cluster details	You can select the **Cluster ID** to view exhaustive	

Capability	Console	
	cluster details like configuration options, persistent application UIs, and logs.	
Searching for clusters	Use a single search field to enter text search queries and to create and apply data filters like "Status = Any active status".	
Finding failed clusters	To search for failed clusters, apply the filter **Status** = **Terminated with errors**.	

Viewing or editing cluster details

Capability	Console	
Viewing the instances in your instance groups and instance fleets, along with scaling, provisioning, resizing, and termination options	View instance options and details in the **Instances** tab. View termination options in the **Properties** tab.	
Viewing app UIs, logs, and configurations (Apache Spark UI, Spark History service, Apache Tez UI, YARN timeline server)	View cluster configurations in the **Configurations** tab. Launch a live, persistent, application UI to see the logs for an application from the **Applications** tab.	
Exporting a cluster to CLI		

Capability	Console	
	Option available from cluster detail and list view Actions menus as "View command for cloning cluster"	

Differences when you work with security configurations

Capability	Console	
Cloning security configurations	✓	
Federated governance using Trino and Apache Ranger	✓	
Using a runtime role to submit work to a cluster[1]	✓	
Authorizing access to EMR File System (EMRFS) data	Amazon S3 access points	
AWS Lake Formation access controls	Runtime roles	

[1] To pass a role during step submission, your cluster must use a security configuration with an IAM permissions policy attached so that the a user can pass only the approved roles and your jobs can access Amazon EMR resources. For more information, see Runtime roles for Amazon EMR steps.

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 with AWS IAM Identity Center with or without trusted identity propagation 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 in EMR Studio.
- 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 with 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 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/.

EMR Studio is also FedRamp compliant. For more information about compliance programs Amazon EMR conforms with, see Compliance validation for Amazon EMR. For more information about additional compliance programs for AWS services, see AWS Services in Scope by Compliance Program.

Amazon SageMaker Unified Studio integrated development environment

Amazon SageMaker Unified Studio provides an integrated development environment (IDE) for your Jupyter notebooks that runs on Amazon EMR on EC2 clusters or using EMR Serverless compute connections. By combining the power of Amazon EMR with the end-to-end workflow capabilities of Amazon SageMaker Unified Studio, teams can streamline data preparation, pipeline development, and ML experimentation in a single environment. Amazon EMR in SageMaker revolutionizes big data processing by supporting open-source frameworks like Apache Spark, Trino, and Apache Flink. Eliminate infrastructure management complexities while scaling analytics workloads effortlessly. To learn more, see Amazon EMR.

Amazon EMR Studio feature history

This table lists updates to the Amazon EMR managed scaling capability.

Release date	Capability
January 5, 2024	Added support for EMR Studio in AWS GovCloud (US-East) and AWS GovCloud (US-West).
November 26, 2023	Added support for trusted identity propagation for EMR Studio with IAM Identity Center authentication.
October 26, 2023	Added ability to create an EMR Serverless application with interactive capability.

Release date	Capability
February 28, 2023	Added AWS KMS customer-managed key support for application log storage for EMR Serverless applications.
February 23, 2023	Added one-click IAM role creation for EMR Serverless job submission. Added ECR lookup for when you select a custom image for EMR Serverless applications.
January 27, 2023	Headless execution notebooks can track the progress of each cell execution with `%execute_notebook` magic.
January 23, 2023	Persistent application have been optimized for faster launch times.

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 a 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 and Define security groups to control EMR Studio network traffic.
- **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 EMR Studio.
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.
3. Start monitoring EMR Studio actions with AWS CloudTrail events. For more information, see Monitor Amazon EMR Studio actions.
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. 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 and Trusted Identity Propagation across applications in the *AWS IAM Identity Center User Guide*.

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**.

Authentication mode	Login method	Description
		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. 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 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.	Define IAM permissions policies that allow certain EMR Studio actions. For native a users, attach the IAM permissions policy to an IAM identity (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 for Amazon EC2 or Amazon EKS.

Authentication mode	User assignment	User authorization
	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.	
IAM Identity Center	For Studios created with `IdCUserAssignment` set to `REQUIRED`, map users to the Studio with a specified session policy. For more information, see Assign a user or group to an EMR Studio. For Studios created with `IdCUserAssignment` set to `OPTIONAL`, any Identity Center user or group can access the Studio.	*Optional:* 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.

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.

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 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. For more information, see Learn EMR Studio workspaces.

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 in EMR Studio
- You can collaborate simultaneously with other users in a Workspace when you enable Workspace collaboration. For more information, see Configure Workspace collaboration in EMR Studio.

- 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.
- 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 EMR Studio workspaces.

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 in EMR Studio.

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 in EMR Studio.

EMR Studio features, requirements, and limits

This topic includes Items to consider when working with Amazon EMR Studio, including considerations about regions and tools, cluster requirements, and technical limitations.

Considerations

Consider the following when you work with EMR Studio:

- EMR Studio is available in the following AWS Regions:
  - US East (Ohio) (us-east-2)

- US East (N. Virginia) (us-east-1)
- US West (N. California) (us-west-1)
- US West (Oregon) (us-west-2)
- Africa (Cape Town) (af-south-1)
- Asia Pacific (Hong Kong) (ap-east-1)
- Asia Pacific (Jakarta) (ap-southeast-3)*
- Asia Pacific (Melbourne) (ap-southeast-4)*
- Asia Pacific (Mumbai) (ap-south-1)
- 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)
- Europe (Frankfurt) (eu-central-1)
- Europe (Ireland) (eu-west-1)
- Europe (London) (eu-west-2)
- Europe (Milan) (eu-south-1)
- Europe (Paris) (eu-west-3)
- Europe (Spain) (eu-south-2)
- Europe (Stockholm) (eu-north-1)
- Europe (Zurich) (eu-central-2)*
- Israel (Tel Aviv) (il-central-1)*
- Middle East (UAE) (me-central-1)*
- South America (São Paulo) (sa-east-1)
- AWS GovCloud (US-East) (gov-us-east-1)
- AWS GovCloud (US-West) (gov-us-west-1)

* The live Spark UI isn't supported in these Regions.

- 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 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 Amazon EMR service role. Session policies aren't supported with these permissions.
- 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.
- There are additional considerations when you use trusted identity propagation with Amazon EMR that also apply to EMR Studio. For more information, see Considerations and limitations for Amazon EMR with the Identity Center integration.
- 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)
```

- To augment the security for the off-console applications that you might use with Amazon EMR, the application hosting domains are registered in the Public Suffix List (PSL). Examples of these hosting domains include the following: `emrstudio-prod.us-east-1.amazonaws.com`, `emrnotebooks-prod.us-east-1.amazonaws.com`, `emrappui-prod.us-east-1.amazonaws.com`. For further security, if you ever need to set sensitive cookies in the default domain name, we recommend that you use cookies with a `__Host-` prefix.

This helps to defend your domain against cross-site request forgery attempts (CSRF). For more information, see the Set-Cookie page in the *Mozilla Developer Network*.

- Amazon EMR Studio Workspaces and Persistent UI endpoints use FIPS 140 validated cryptographic modules for encryption-in-transit, which enables easier adoption of the service for regulated workloads. For additional context on Persistent UI endpoints, see View persistent application user interfaces in Amazon EMR. For additional context regarding notebooks, see Amazon EMR Notebooks overview.

Known issues

- An EMR Studio that uses IAM Identity Center with trusted identity propagation enabled can only associate with EMR clusters that also use trusted identity propagation.
- 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 Amazon 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 (Amazon EMR 5.x series) and 6.2.0 (Amazon EMR 6.x series) and higher.
- When you Debug Amazon EMR running on Amazon EC2 jobs, 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 primary 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 Amazon EMR cluster primary node using SSH, 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, 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. For more information about auto-termination, see Using an auto-termination policy for Amazon EMR cluster cleanup.
- 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.
- Using Amazon EMR Studio Workspaces as the root user in an AWS account causes a `403: Forbidden` error. This is because the Jupyter Enterprise Gateway configuration in Amazon EMR doesn't allow access to the root user. We recommend that you don't use the root user for your everyday tasks. For other authentication options, see AWS Identity and Access Management for Amazon EMR.

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 primary nodes
- Clusters that use Amazon EC2 instances based on AWS Graviton2 for Amazon EMR 6.x releases lower than 6.9.0, and 5.x releases lower than 5.36.1

The following features aren't supported from a Studio that uses trusted identity propagation:

- Creating EMR clusters without a template.
- Using EMR Serverless applications.
- Launching Amazon EMR on EKS clusters.
- Using a runtime role.
- Enabling SQL Explorer or Workspace collaboration.

Service limits for EMR Studio

The following table displays service limits for EMR Studio.

Item	Limit
EMR Studios	Maximum of 100 per AWS account
Subnets	Maximum of 5 associated with each EMR Studio
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 for EMR Studio

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.
- 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 Amazon EMR on EKS with EMR Studio, choose the same VPC as your Amazon EKS cluster worker nodes.

Amazon EMR cluster requirements

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 (Amazon EMR 5.x series) or 6.2.0 (Amazon 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 compute to an EMR Studio Workspace.
- 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 primary node of a cluster. Doing so relieves the primary 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 Planning and configuring instance fleets for your Amazon EMR cluster.
- 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.
- 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 managed scaling in Amazon EMR.

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 and Control network traffic with security groups for your Amazon EMR cluster.

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 Amazon 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 Amazon 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
- Set up an EMR Studio
- Monitor, update and delete Amazon EMR Studio resources
- Encrypting EMR Studio workspace notebooks and files
- Define security groups to control EMR Studio network traffic
- Create AWS CloudFormation templates for Amazon EMR Studio
- Establish access and permissions for Git-based repositories
- Optimize Spark jobs in EMR Studio

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.

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:CreateApplication", "sso:PutApplicationAuthenticationMethod", "sso:PutApplicationGrant", "sso:PutApplicationAccessScope", "sso:PutApplicationAssignmentConfiguration", "iam:PassRole"`
Describe a Studio	`"elasticmapreduce:DescribeStudio", "sso:GetManagedApplicationInstance"`
List Studios	`"elasticmapreduce:ListStudios"`
Delete a Studio	`"elasticmapreduce:DeleteStudio", "sso:DeleteApplication",`

Operation	Permissions
	`"sso:DeleteApplicationAuthentica tionMethod", "sso:DeleteApplicationAccessScope", "sso:DeleteApplicationGrant"`
Additional permissions required when you use IAM Identity Center mode	
Assign users or groups to a Studio	`"elasticmapreduce:CreateStudioSessio nMapping", "sso:GetProfile", "sso:ListDirectoryAssociations", "sso:ListProfiles", "sso:AssociateProfile", "sso-directory:SearchUsers", "sso-directory:SearchGroups", "sso-directory:DescribeUser", "sso-directory:DescribeGroup", "sso:ListInstances", "sso:CreateApplicationAssignment", "sso:DescribeInstance", "organizations:DescribeOrga nization", "organizations:ListDelegatedAdmini strators", "sso:CreateInstance", "sso:DescribeRegisteredRegions", "sso:GetSharedSsoConfiguration", "iam:ListPolicies"`
Retrieve Studio assignment details for a specific user or group	`"sso-directory:SearchUsers", "sso-directory:SearchGroups", "sso-directory:DescribeUser", "sso-directory:DescribeGroup", "sso:DescribeApplication", "elasticmapreduce:GetStudioSessio nMapping"`

Operation	Permissions
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:DescribeApplication", "sso:DescribeInstance", "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:DescribeApplication", "sso:DescribeInstance", "sso:ListProfiles", "sso:DisassociateProfile", "sso:DeleteApplicationAssignment", "sso:ListApplicationAssignments"`

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.

   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 and EMR Studio user role.

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

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticmapreduce:*:123456789012:studio/*"
      ],
      "Action": [
        "elasticmapreduce:CreateStudio",
        "elasticmapreduce:DescribeStudio",
        "elasticmapreduce:DeleteStudio"
      ],
      "Sid": "AllowELASTICMAPREDUCECreatestudio"
    },
    {
      "Effect": "Allow",
      "Resource": [
        "*"
      ],
      "Action": [
        "elasticmapreduce:ListStudios"
      ],
      "Sid": "AllowELASTICMAPREDUCEListstudios"
    },
    {
      "Effect": "Allow",
      "Resource": [
        "arn:aws:iam::123456789012:role/EMRStudioServiceRole"
      ],
      "Action": [
```

```
        "iam:PassRole"
      ],
      "Sid": "AllowIAMPassrole"
    }
  ]
}
```

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

> **Note**
>
> Identity Center and 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.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Resource": [
        "arn:aws:elasticmapreduce:*:123456789012:studio/*"
      ],
      "Action": [
        "elasticmapreduce:CreateStudio",
        "elasticmapreduce:DescribeStudio",
        "elasticmapreduce:DeleteStudio",
        "elasticmapreduce:CreateStudioSessionMapping",
        "elasticmapreduce:GetStudioSessionMapping",
        "elasticmapreduce:UpdateStudioSessionMapping",
        "elasticmapreduce:DeleteStudioSessionMapping"
      ],
      "Sid": "AllowELASTICMAPREDUCECreatestudio"
```

```
    },
    {
      "Effect": "Allow",
      "Resource": [
        "*"
      ],
      "Action": [
        "elasticmapreduce:ListStudios",
        "elasticmapreduce:ListStudioSessionMappings"
      ],
      "Sid": "AllowELASTICMAPREDUCEListstudios"
    },
    {
      "Effect": "Allow",
      "Resource": [
        "arn:aws:iam::123456789012:role/EMRStudio-SvcRole",
        "arn:aws:iam::123456789012:role/EMRStudio-User-Role"
      ],
      "Action": [
        "iam:PassRole"
      ],
      "Sid": "AllowIAMPassrole"
    },
    {
      "Effect": "Allow",
      "Resource": [
        "*"
      ],
      "Action": [
        "sso:CreateApplication",
        "sso:PutApplicationAuthenticationMethod",
        "sso:PutApplicationGrant",
        "sso:PutApplicationAccessScope",
        "sso:PutApplicationAssignmentConfiguration",
        "sso:DescribeApplication",
        "sso:DeleteApplication",
        "sso:DeleteApplicationAuthenticationMethod",
        "sso:DeleteApplicationAccessScope",
        "sso:DeleteApplicationGrant",
        "sso:ListInstances",
        "sso:CreateApplicationAssignment",
        "sso:DeleteApplicationAssignment",
        "sso:ListApplicationAssignments",
        "sso:DescribeInstance",
```

```
        "sso:AssociateProfile",
        "sso:DisassociateProfile",
        "sso:GetProfile",
        "sso:ListDirectoryAssociations",
        "sso:ListProfiles",
        "sso-directory:SearchUsers",
        "sso-directory:SearchGroups",
        "sso-directory:DescribeUser",
        "sso-directory:DescribeGroup",
        "organizations:DescribeOrganization",
        "organizations:ListDelegatedAdministrators",
        "sso:CreateInstance",
        "sso:DescribeRegisteredRegions",
        "sso:GetSharedSsoConfiguration",
        "iam:ListPolicies"
      ],
      "Sid": "AllowSSOCreateapplication"
    }
  ]
}
```

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

Set up an EMR Studio

Complete the following steps to set up an 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 Before you set up Amazon EMR.
- Permissions to create and manage an EMR Studio. For more information, see the section called "Administrator permissions to create an EMR Studio".

- 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*.
- If you want to attach to an Amazon EMR on EC2 or Amazon EMR on EKS cluster, or use Git repositories, you need an Amazon Virtual Private Cloud (VPC) for the Studio, and a maximum of five subnets. You don't need a VPC to use EMR Studio with EMR Serverless. For tips on how to configure networking, see VPC and subnet best practices for EMR Studio.

To set up an EMR Studio

1. Choose an authentication mode for Amazon EMR Studio
2. Create the following Studio resources.
   - Create an EMR Studio service role
   - Configure EMR Studio user permissions for Amazon EC2 or Amazon EKS
   - (Optional) Define security groups to control EMR Studio network traffic.
3. Create an EMR Studio
4. Assign a user or group to an EMR Studio

After you complete the setup steps, you can Use an Amazon EMR Studio.

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. 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.

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, which offers the following benefits:

If you are...	We recommend...
	• 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, 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 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.
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 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 instructi ons, 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.

If you use...	Do this...
	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.

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 on Amazon EC2 or Amazon EKS

1. Follow the instructions in Creating a role to delegate permissions to an AWS service to create the service role with 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.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": "arn:aws:iam::123456789012:role/EMRStudioServiceRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:elasticmapreduce:*:123456789012:*"
        }
      },
      "Sid": "AllowSTSAssumerole"
    }
  ]
```

```
}
```

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.

> **Important**
>
> - For Amazon EC2 tag-based access control with to work with EMR Studio, you must set access for the `ModifyNetworkInterfaceAttribute` API as shown the following policy.
> - For EMR Studio to work with the service role, you must not change the following statements: `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.

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

JSON

```
{
  "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"
```

4. If you want to control access to Git secrets at user level, add tag-based permissions to `secretsmanager:GetSecretValue` in the EMR Studio **user role policy**, and remove permissions to `secretsmanager:GetSecretValue` policy from the EMR Studio **service role policy**. For more information on setting fine-grained user permissions, see Create permissions policies for EMR Studio users.

Minimum service role for EMR Serverless

If you want to run interactive workloads with EMR Serverless through EMR Studio notebooks, use the same trust policy that you use to set up EMR Studio in the previous section, How to create a service role for EMR Studio on Amazon EC2 or Amazon EKS.

For your IAM policy, the minimum viable policy has permissions as follows. Update *bucket-name* with the name of the bucket that you plan to use when you configure your EMR Studio and Workspace. EMR Studio uses the bucket back up the Workspaces and notebook files in your Studio.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ObjectActions",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::bucket-name/*"
      ]
    },
    {
      "Sid": "BucketActions",
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:GetEncryptionConfiguration"
      ],
      "Resource": [
```

```
        "arn:aws:s3:::bucket-name"
      ]
    }
  ]
}
```

If you plan to use an encrypted Amazon S3 bucket, add the following permissions on your policy:

```
"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"`

Operation	Actions
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 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"`
	
	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",`
	`"sso:DescribeApplication",`
	`"sso:DescribeInstance"`

Operation	Actions

Encrypt EMR Studio workspace notebooks and files using customer managed keys (CMK) with AWS Key Management Service	`"kms:Decrypt",`
	`"kms:GenerateDataKey",`
	`"kms:ReEncryptFrom",`
	`"kms:ReEncryptTo",`
	`"kms:DescribeKey"`

Configure EMR Studio user permissions for Amazon EC2 or Amazon EKS

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 in How Amazon EMR Studio works.

> **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.

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.

   When you create the role, use the following trust relationship policy.

   JSON

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

```
        "Effect": "Allow",
        "Action": [
          "sts:AssumeRole",
          "sts:SetContext"
        ],
        "Resource": "arn:aws:iam::123456789012:role/EMRStudioServiceRole",
        "Sid": "AllowSTSAssumerole"
      }
    ]
}
```

2. Remove the default role permissions and policies.
3. Before you assign users and groups to a Studio, attach your EMR Studio session policies to the user role. For instructions on how to create session policies, see Create permissions policies for EMR Studio users.

Create permissions policies for EMR Studio users

Refer to the following sections to create permissions policies for EMR Studio.

Topics

- Create the permissions policies
- Set ownership for Workspace collaboration
- Create user-level Git secrets policy
- Attach the permissions policy to your IAM identity

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

Create the permissions policies

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

For a breakdown of each Studio operation that a user might perform, and the minimum IAM actions that are required to perform each operation, see AWS Identity and Access Management permissions for EMR Studio users. For steps to create the policies, see Creating IAM policies in the *IAM User Guide*.

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 adds the `creatorUserId` tag to Workspaces created after November 16, 2021. To restrict who can configure collaboration for workspaces that you created before this date, 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 in the *IAM User Guide*.

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

Create user-level Git secrets policy

Topics

- To use user-level permissions
- To transition from service-level permissions to user-level permissions
- To use service-level permissions

To use user-level permissions

EMR Studio automatically adds the `for-use-with-amazon-emr-managed-user-policies` tag when it creates Git secrets. If you want to control access to Git secrets at the user level, add tag-based permissions to the EMR Studio **user role policy** with `secretsmanager:GetSecretValue` as shown in the To transition from service-level permissions to user-level permissions section below.

If you have existing permissions for `secretsmanager:GetSecretValue` in the EMR Studio **service role policy**, you should remove those permissions.

To transition from service-level permissions to user-level permissions

> **Note**
>
> The `for-use-with-amazon-emr-managed-user-policies` tag ensures that the permissions from **Step 1** below grant the creator of the workspace access to the Git secret. However, if you linked Git repositories before September 1, 2023, then the corresponding Git secrets will be denied access because they don't have the `for-use-with-amazon-emr-managed-user-policies` tag applied. To apply user-level permissions, you must recreate the old secrets from JupyterLab and link the appropriate Git repositories again. For more information about policy variables, see IAM policy elements: Variables and tags in the *IAM User Guide*.

1. Add the following permissions to the the EMR Studio **user role policy**. It uses the `for-use-with-amazon-emr-managed-user-policies` key with value `"${aws:userid}"`.

```
{
   "Sid": "AllowSecretsManagerReadOnlyActionsWithEMRTags",
    "Effect": "Allow",
    "Action": "secretsmanager:GetSecretValue",
    "Resource": "arn:aws:secretsmanager:*:*:secret:*",
    "Condition": {
        "StringEquals": {
            "secretsmanager:ResourceTag/for-use-with-amazon-emr-managed-user-
policies": "${aws:userid}"
        }
    }
}
```

2. If present, remove the following permission from the EMR Studio **service role policy**. Because the service role policy applies to all secrets defined by each user, you only need to do this one time.

```
{
    "Sid": "AllowSecretsManagerReadOnlyActionsWithEMRTags",
    "Effect": "Allow",
    "Action": [
        "secretsmanager:GetSecretValue"
     ],
    "Resource": "arn:aws:secretsmanager:*:*:secret:*",
    "Condition": {
        "StringEquals": {
            "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
        }
    }
}
```

To use service-level permissions

As of September 1, 2023, EMR Studio automatically adds the `for-use-with-amazon-emr-managed-user-policies` tag for user-level access control. Because this is an added capability, you can continue to use service-level access that's available through the `GetSecretValue` permission in the EMR Studio service role.

For secrets created before September 1, 2023, EMR Studio didn't add the `for-use-with-amazon-emr-managed-user-policies` tag. To keep using service-level permissions, simply retain your existing EMR Studio service role and user role permissions. However, to restrict who can access an individual secret, we recommend that you follow the steps in To use user-level permissions to manually add the `for-use-with-amazon-emr-managed-user-policies` tag to your secrets, and then use tag-based access control in your user permissions policies.

For more information about policy variables, see IAM policy elements: Variables and tags in the *IAM User Guide*.

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 permissio ns 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.

Example user policies

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

Basic policy

> **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.

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.

JSON

```
{
  "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-111122223333>-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": [
        "*"
      ]
    }
  ]
}
```

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

Intermediate policy

> **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.

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.

JSON

```
{
  "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:DescribeJobRun",
        "emr-containers:ListJobRuns"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    },
    {
      "Sid": "AllowRetrievingManagedEndpointCredentials",
      "Effect": "Allow",
      "Action": [
        "emr-containers:GetManagedEndpointSessionCredentials"
```

```
      ],
      "Resource": [
        "arn:aws:emr-containers:us-west-1:123456789012:/virtualclusters/virtual-
cluster-id/endpoints/managed-endpoint-id"
      ],
      "Condition": {
        "StringEquals": {
          "emr-containers:ExecutionRoleArn": [
            "arn:aws:iam::123456789012:role/emr-on-eks-execution-role"
          ]
        }
      }
    },
    {
      "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-123456789012-us-east-1/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": [
        "*"
      ]
    },
    {
      "Sid": "AllowServerlessActions",
      "Action": [
        "emr-serverless:CreateApplication",
        "emr-serverless:UpdateApplication",
        "emr-serverless:DeleteApplication",
        "emr-serverless:ListApplications",
        "emr-serverless:GetApplication",
        "emr-serverless:StartApplication",
        "emr-serverless:StopApplication",
        "emr-serverless:StartJobRun",
        "emr-serverless:CancelJobRun",
        "emr-serverless:ListJobRuns",
        "emr-serverless:GetJobRun",
        "emr-serverless:GetDashboardForJobRun",
        "emr-serverless:AccessInteractiveEndpoints"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    },
    {
      "Sid": "AllowPassingRuntimeRoleForRunningServerlessJob",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/serverless-runtime-role"
      ],
      "Effect": "Allow"
    }
  ]
}
```

The following advanced 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.

Advanced policy

> **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.

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.

JSON

```
{
  "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:DescribeJobRun",
        "emr-containers:ListJobRuns"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    },
    {
      "Sid": "AllowRetrievingManagedEndpointCredentials",
      "Effect": "Allow",
      "Action": [
        "emr-containers:GetManagedEndpointSessionCredentials"
      ],
      "Resource": [
        "arn:aws:emr-containers:*:123456789012:/virtualclusters/virtual-cluster-
id/endpoints/managed-endpoint-id"
      ],
      "Condition": {
        "StringEquals": {
          "emr-containers:ExecutionRoleArn": [
            "arn:aws:iam::123456789012:role/emr-on-eks-execution-role"
          ]
        }
```

```
        }
      },
      {
        "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_V2",
        "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-123456789012-us-east-1/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": "SageMakerDataWranglerForEMRStudio",
        "Effect": "Allow",
        "Action": [
          "sagemaker:CreatePresignedDomainUrl",
          "sagemaker:DescribeDomain",
          "sagemaker:ListDomains",
          "sagemaker:ListUserProfiles"
        ],
        "Resource": [
          "*"
        ]
      },
      {
```

```
        "Sid": "DescribeNetwork",
        "Effect": "Allow",
        "Action": [
          "ec2:DescribeVpcs",
          "ec2:DescribeSubnets",
          "ec2:DescribeSecurityGroups"
        ],
        "Resource": [
          "*"
        ]
      },
      {
        "Sid": "ListIAMRoles",
        "Effect": "Allow",
        "Action": [
          "iam:ListRoles"
        ],
        "Resource": [
          "*"
        ]
      },
      {
        "Sid": "AllowServerlessActions",
        "Action": [
          "emr-serverless:CreateApplication",
          "emr-serverless:UpdateApplication",
          "emr-serverless:DeleteApplication",
          "emr-serverless:ListApplications",
          "emr-serverless:GetApplication",
          "emr-serverless:StartApplication",
          "emr-serverless:StopApplication",
          "emr-serverless:StartJobRun",
          "emr-serverless:CancelJobRun",
          "emr-serverless:ListJobRuns",
          "emr-serverless:GetJobRun",
          "emr-serverless:GetDashboardForJobRun",
          "emr-serverless:AccessInteractiveEndpoints"
        ],
        "Resource": [
          "*"
        ],
        "Effect": "Allow"
      },
      {
```

```
        "Sid": "AllowPassingRuntimeRoleForRunningServerlessJob",
        "Action": [
          "iam:PassRole"
        ],
        "Resource": [
          "arn:aws:iam::*:role/serverless-runtime-role"
        ],
        "Effect": "Allow"
      },
      {
        "Sid": "AllowCodeWhisperer",
        "Effect": "Allow",
        "Action": [
          "codewhisperer:GenerateRecommendations"
        ],
        "Resource": [
          "*"
        ]
      },
      {
        "Sid": "AllowAthenaSQL",
        "Action": [
          "athena:StartQueryExecution",
          "athena:StopQueryExecution",
          "athena:GetQueryExecution",
          "athena:GetQueryRuntimeStatistics",
          "athena:GetQueryResults",
          "athena:ListQueryExecutions",
          "athena:BatchGetQueryExecution",
          "athena:GetNamedQuery",
          "athena:ListNamedQueries",
          "athena:BatchGetNamedQuery",
          "athena:UpdateNamedQuery",
          "athena:DeleteNamedQuery",
          "athena:ListDataCatalogs",
          "athena:GetDataCatalog",
          "athena:ListDatabases",
          "athena:GetDatabase",
          "athena:ListTableMetadata",
          "athena:GetTableMetadata",
          "athena:ListWorkGroups",
          "athena:GetWorkGroup",
          "athena:CreateNamedQuery",
          "athena:GetPreparedStatement",
```

```
        "glue:CreateDatabase",
        "glue:DeleteDatabase",
        "glue:GetDatabase",
        "glue:GetDatabases",
        "glue:UpdateDatabase",
        "glue:CreateTable",
        "glue:DeleteTable",
        "glue:BatchDeleteTable",
        "glue:UpdateTable",
        "glue:GetTable",
        "glue:GetTables",
        "glue:BatchCreatePartition",
        "glue:CreatePartition",
        "glue:DeletePartition",
        "glue:BatchDeletePartition",
        "glue:UpdatePartition",
        "glue:GetPartition",
        "glue:GetPartitions",
        "glue:BatchGetPartition",
        "kms:ListAliases",
        "kms:ListKeys",
        "kms:DescribeKey",
        "lakeformation:GetDataAccess",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads",
        "s3:ListMultipartUploadParts",
        "s3:AbortMultipartUpload",
        "s3:PutObject",
        "s3:PutBucketPublicAccessBlock",
        "s3:ListAllMyBuckets"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

The following user policy contains the minimum user permissions that are required to use an EMR Serverless interactive application with EMR Studio Workspaces.

EMR Serverless interactive policy

In this example policy that has user permissions for EMR Serverless interactive applications with EMR Studio, replace the placeholders for *serverless-runtime-role* and *emr-studio-service-role* with your correct EMR Studio service role and EMR Serverless runtime role.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowServerlessActions",
      "Action": [
        "emr-serverless:CreateApplication",
        "emr-serverless:UpdateApplication",
        "emr-serverless:DeleteApplication",
        "emr-serverless:ListApplications",
        "emr-serverless:GetApplication",
        "emr-serverless:StartApplication",
        "emr-serverless:StopApplication",
        "emr-serverless:StartJobRun",
        "emr-serverless:CancelJobRun",
        "emr-serverless:ListJobRuns",
        "emr-serverless:GetJobRun",
        "emr-serverless:GetDashboardForJobRun",
        "emr-serverless:AccessInteractiveEndpoints"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    },
    {
      "Sid": "AllowEMRBasicActions",
      "Action": [
        "elasticmapreduce:CreateEditor",
        "elasticmapreduce:DescribeEditor",
        "elasticmapreduce:ListEditors",
        "elasticmapreduce:UpdateStudio",
        "elasticmapreduce:StartEditor",
        "elasticmapreduce:StopEditor",
        "elasticmapreduce:DeleteEditor",
```

```
        "elasticmapreduce:OpenEditorInConsole",
        "elasticmapreduce:AttachEditor",
        "elasticmapreduce:DetachEditor",
        "elasticmapreduce:CreateStudio",
        "elasticmapreduce:DescribeStudio",
        "elasticmapreduce:DeleteStudio",
        "elasticmapreduce:ListStudios",
        "elasticmapreduce:CreateStudioPresignedUrl"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    },
    {
      "Sid": "AllowPassingRuntimeRoleForRunningEMRServerlessJob",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/serverless-runtime-role"
      ],
      "Effect": "Allow"
    },
    {
      "Sid": "AllowPassingServiceRoleForWorkspaceCreation",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/emr-studio-service-role"
      ],
      "Effect": "Allow"
    },
    {
      "Sid": "AllowS3ListAndGetPermissions",
      "Action": [
        "s3:ListAllMyBuckets",
        "s3:ListBucket",
        "s3:GetBucketLocation",
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::*"
```

```
        ],
        "Effect": "Allow"
      },
      {
        "Sid": "DescribeNetwork",
        "Effect": "Allow",
        "Action": [
          "ec2:DescribeVpcs",
          "ec2:DescribeSubnets",
          "ec2:DescribeSecurityGroups"
        ],
        "Resource": [
          "*"
        ]
      },
      {
        "Sid": "ListIAMRoles",
        "Effect": "Allow",
        "Action": [
          "iam:ListRoles"
        ],
        "Resource": [
          "*"
        ]
      }
    ]
}
```

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 user role 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.

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

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

Action	Basic	Intermediate	Advanced	Associated actions
Attach or detach EMR Serverless applications that are associated with the Workspace	No	Yes	Yes	`"elasticmapreduce:AttachEditor", "elasticmapreduce:DetachEditor", "emr-serverless:GetApplication", "emr-serverless:StartApplication", "emr-serverless:ListApplications", "emr-serverless:GetDashboardForJobRun", "emr-serverless:AccessInteractiveEndpoints", "iam:PassRole"` The `PassRole` permission is required to pass the EMR Serverless job runtime role. For more information, see Job runtime roles in the *Amazon EMR Serverless User Guide*.

Action	Basic	Intermediate	Advanced	Associated actions
Debug Amazon EMR on EC2 jobs with persistent application user interfaces	Yes	Yes	Yes	`"elasticmapreduce:CreatePersistentAppUI", "elasticmapreduce:DescribePersistentAppUI", "elasticmapreduce:GetPersistentAppUIPresignedURL", "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:GetOnClusterAppUIPresignedURL"`
Debug Amazon EMR on EKS job runs using the Spark History Server	Yes	Yes	Yes	`"elasticmapreduce:CreatePersistentAppUI", "elasticmapreduce:DescribePersistentAppUI", "elasticmapreduce:GetPersistentAppUIPresignedURL", "emr-containers:ListVirtualClusters", "emr-containers:DescribeVirtualCluster", "emr-containers:ListJobRuns", "emr-containers:DescribeJobRun", "s3:ListBucket", "s3:GetObject"`

Action	Basic	Intermediate	Advanced	Associated actions
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"`
Provide a cluster configuration to create new clusters.	No	No	Yes	`"elasticmapreduce:RunJobFlow", "iam:PassRole", "elasticmapreduce:ListClusters", "elasticmapreduce:DescribeCluster"`
Assign a user to a Studio when you use IAM authentication mode.	No	No	No	`"elasticmapreduce:CreateStudioPresignedUrl"`

Action	Basic	Intermediate	Advanced	Associated actions
Describe network objects.	Yes	Yes	Yes	JSON `{ "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	JSON `{ "Version": "2012-10-17", "Statement": [ { "Sid": "ListIAMRoles", "Effect": "Allow", "Action": [ "iam:ListRoles" ], "Resource": [ "*" ] } ] }`
Connect to EMR Studio from Amazon SageMaker AI Studio and use the Data Wrangler visual interface.	No	No	Yes	`"sagemaker:CreatePresignedDomainUrl", "sagemaker:DescribeDomain", "sagemaker:ListDomains", "sagemaker:ListUserProfiles"`
Use Amazon CodeWhisperer in your EMR Studio.	No	No	Yes	`"codewhisperer:GenerateRecommendations"`

Action	Basic	Intermediate	Advanced	Associated actions
Access Amazon Athena SQL editor from your EMR Studio. This list might not include all of the permissions that you need to use all Athena features. For the most up-to-date list, see the Athena full access policy.	No	No	Yes	`"athena:StartQueryExecution", "athena:StopQueryExecution", "athena:GetQueryExecution", "athena:GetQueryRuntimeStatistics", "athena:GetQueryResults", "athena:ListQueryExecutions", "athena:BatchGetQueryExecution", "athena:GetNamedQuery", "athena:ListNamedQueries", "athena:BatchGetNamedQuery", "athena:UpdateNamedQuery", "athena:DeleteNamedQuery", "athena:ListDataCatalogs", "athena:GetDataCatalog", "athena:ListDatabases", "athena:GetDatabase", "athena:ListTableMetadata", "athena:GetTableMetadata", "athena:ListWorkGroups", "athena:GetWorkGroup", "athena:CreateNamedQuery", "athena:GetPreparedStatement", "glue:CreateDatabase", "glue:DeleteDatabase", "glue:GetDatabase",`

Action	Basic	Intermediate	Advanced	Associated actions
				"glue:GetDatabases", "glue:UpdateDatabase", "glue:CreateTable", "glue:DeleteTable", "glue:BatchDeleteTable", "glue:UpdateTable", "glue:GetTable", "glue:GetTables", "glue:BatchCreatePartition", "glue:CreatePartition", "glue:DeletePartition", "glue:BatchDeletePartition", "glue:UpdatePartition", "glue:GetPartition", "glue:GetPartitions", "glue:BatchGetPartition", "kms:ListAliases", "kms:ListKeys", "kms:DescribeKey", "lakeformation:GetDataAccess", "s3:GetBucketLocation", "s3:GetBucketLocation", "s3:GetObject", "s3:ListBucket", "s3:ListBucketMultipartUploads", "s3:ListMultipartUploadParts", "s3:AbortMultipartUpload", "s3:PutObject", "s3:PutBucketPublicAccessBlock", "s3:ListAllMyBuckets"

Create an EMR Studio

You can create an EMR Studio for your team with 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 EMR Studio.

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**.

Amazon EMR provides you with a simple console experience to create a Studio, so you can quickly get started with the default settings. to run interactive workloads or batch jobs with the default settings. Creating a EMR Studio also creates an EMR Serverless application ready for your interactive jobs.

If you want full control over your Studio's settings, you can choose **Custom**, which lets you configure all of the additional settings.

Interactive workloads

To create a EMR Studio for interactive workloads

1. Open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR Studio** on the left navigation, choose **Getting started**. You can also create a new Studio from the **Studios** page.
3. Amazon EMR provides default settings for you if you're creating a EMR Studio for interactive workloads, but you can edit these settings. Configurable settings include the EMR Studio's name, the S3 location for your Workspace, the service role to use, the Workspace(s) you want to use, EMR Serverless application name, and the associated runtime role.

4. Choose **Create Studio and launch Workspace** 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**. Your Workspace opens in a new tab in your browser.

Batch jobs

To create a EMR Studio for interactive workloads

1. Open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR Studio** on the left navigation, choose **Getting started**. You can also create a new Studio from the **Studios** page.
3. Amazon EMR provides default settings for you if you're creating a EMR Studio for batch jobs, but you can edit these settings. Configurable settings include the EMR Studio's name, EMR Serverless application name, and the associated runtime role.
4. Choose **Create Studio and launch Workspace** 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**. Your EMR Studio opens in a new tab in your browser.

Custom settings

To create a EMR Studio with custom settings

1. Open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR Studio** on the left navigation, choose **Getting started**. You can also create a new Studio from the **Studios** page.
3. Choose **Create a Studio** to open the **Create a Studio** page.
4. Enter a **Studio name**.
5. Choose to create a new S3 bucket or use an existing location.
6. Choose the Workspace to add to the Studio. You can add up to 3 Workspaces.
7. 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.

If you use...	Do this...
IAM authentication or federation	The default authentication method is **AWS Identity and Access Management (IAM)**. At the bottom of the screen, you can also add tags to give specific users access to the Studio as described in Assign a user or group to an EMR Studio. 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.

If you use...	Do this...
IAM Identity Center authentication	Select your EMR Studio **Service Role** and **User Role**. For more information, see Create an EMR Studio service role and Create an EMR Studio user role for IAM Identity Center authentication mode. When you use **IAM Identity Center (formerly AWS Single Sign On)** authentic ation for the Studio, you can choose to streamline the sign-on experience for users with the **Enable trusted identity propagation** option. With trusted identity propagation, users can log in with their Identity Center credentials and have their identities propagated to downstream AWS services when they use the Studio. In the **Application access** section, you can also specify whether all users and groups in your Identity Center should have access to the Studio, or if only assigned users and groups that you choose can access the Studio. For more information, see Integrate Amazon EMR with AWS IAM Identity Center, and also Trusted Identity Propagati on across applications in the *AWS IAM Identity Center User Guide*.

8. For VPC, choose an Amazon Virtual Private Cloud (**VPC**) for the Studio from the dropdown list.
9. 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.

10. 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.

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.

11. Add tags to your Studio and other resources. For more information about tags, see Tag clusters.
12. Choose **Create Studio and launch Workspace** 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.

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 – Create an EMR Studio that uses IAM for authentication

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.

```
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> \
--user-role 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> \
--idp-auth-url <https://EXAMPLE/login/> \
--idp-relay-state-parameter-name <example-RelayState>
```

Example – Create an EMR Studio that uses Identity Center for authentication

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.

```
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>
--trusted-identity-propagation-enabled \
--idc-user-assignment OPTIONAL \
--idc-instance-arn <iam-identity-center-instance-arn>
```

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.

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>`

Identity provider	Parameter	Authentication URL
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>*
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.

Assign a user or group to an EMR Studio

IAM

When you use Set up IAM authentication mode for Amazon EMR Studio, 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 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.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreateStudioPresignedUrl",
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:CreateStudioPresignedUrl"
      ],
      "Resource": [
        "arn:aws:elasticmapreduce:us-east-1:123456789012: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 a users and roles using tags or Controlling access to AWS resources using tags.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreateStudioPresignedUrl",
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:CreateStudioPresignedUrl"
      ],
      "Resource": [
        "arn:aws:elasticmapreduce:*:123456789012: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>*.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:CreateStudioPresignedUrl"
      ],
      "Resource": [
        "arn:aws:elasticmapreduce:us-east-1:123456789012:studio/studio-name"
      ],
      "Condition": {
        "StringLike": {
          "aws:SourceIdentity": "example-source-identity"
        }
      },
      "Sid": "AllowELASTICMAPREDUCECreatestudiopresignedurl"
    }
  ]
}
```

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 Amazon EMR console

1. Navigate to the new Amazon EMR console and select **Switch to the old console** from the side navigation. For more information on what to expect when you switch to the old console, see Using the old console.
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 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.
- **`--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.

```
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.

IAM Identity Center

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

1. Navigate to the new Amazon EMR console and select **Switch to the old console** from the side navigation. For more information on what to expect when you switch to the old console, see Using the old console.
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.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreateStudioPresignedUrl",
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:CreateStudioPresignedUrl"
      ],
      "Resource": [
        "arn:aws:elasticmapreduce:us-east-1:123456789012:studio/*"
      ],
      "Condition": {
        "StringEquals": {
          "elasticmapreduce:ResourceTag/Team": [
            "Data Analytics",
```

```
            "Quality Assurance"
          ]
        }
      }
    }
  ]
}
```

IAM Identity Center

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

1. Navigate to the new Amazon EMR console and select **Switch to the old console** from the side navigation. For more information on what to expect when you switch to the old console, see Using the old console.
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> \
```

Monitor, update and delete Amazon EMR Studio resources

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.

View Studio details

Console

To view details about an EMR Studio with the new console

1. Open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR Studio** on the left navigation, choose **Studios**.
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 a 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: a user Calls the DescribeStudio API

The following is an example AWS CloudTrail event that is created when a 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":"AIDXXXXXXXXXXXXXXXXX",
      "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-9O5XXXXXXXXXXXXXXXXXXXX"
   },
   "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 primary 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 primary 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. You can also connect to the primary node of the cluster using SSH to view application web interfaces. For more information, see View web interfaces hosted on Amazon EMR clusters.

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. Navigate to the new Amazon EMR console and select **Switch to the old console** from the side navigation. For more information on what to expect when you switch to the old console, see Using the old console.
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

From the Workspace UI

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. If you want to remove the notebook files that were associated with the deleted Workspace from Amazon S3, follow the instructions for Deleting objects in the *Amazon Simple Storage Service Console User Guide*. If you did not create the Studio, consult your Studio administrator to determine the Amazon S3 backup location for the deleted Workspace.

From the Workspaces list

Delete a Workspace and its associated backup files from the Workspaces list

1. Navigate to the **Workspace**s list in the console.
2. Select the Workspace that you want to delete from the list and then choose **Actions**.
3. Choose **Delete**.
4. If you want to remove the notebook files that were associated with the deleted Workspace from Amazon S3, follow the instructions for Deleting objects in the *Amazon Simple Storage Service Console User Guide*. 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 with the new console

1. Open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR Studio** on the left navigation, choose **Studios**.
3. Select the Studio from the **Studios** list with the toggle to the left of the Studio name . Choose **Delete**.

Old console

To delete an EMR Studio with the old console

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

Encrypting EMR Studio workspace notebooks and files

In EMR Studio, you can create and configure different workspaces to organize and run notebooks. These workspaces store notebooks and related files in your specified Amazon S3 bucket. By default, these files are encrypted with Amazon S3-managed keys (SSE-S3) with server-side encryption as the base level of encryption. You can also choose to use customer managed KMS keys (SSE-KMS) to encrypt your files. You can do so by using the Amazon EMR management console or through the AWS CLI and AWS SDK when creating an EMR Studio.

EMR Studio workspace storage encryption is available in all the Regions where EMR Studio is available.

Prerequisites

Before you can encrypt EMR Studio workspace notebook and files, you must use AWS Key Management Service to create a symmetric customer manager key (CMK) in the same AWS account and Region as your EMR Studio.

The resource policy of your AWS KMS must have the necessary access permissions for your EMR Studio's service role. The following is a sample IAM policy granting minimum access permissions for EMR Studio Workspace storage encryption:

```
{
    "Sid": "AllowEMRStudioServiceRoleAccess",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::<ACCOUNT_ID>:role/<ROLE_NAME>"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey",
        "kms:ReEncryptFrom",
        "kms:ReEncryptTo",
        "kms:DescribeKey"
    ],
```

```
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "kms:CallerAccount": "<ACCOUNT_ID>",
            "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::<S3_BUCKET_NAME>",
            "kms:ViaService": "s3.<AWS_REGION>.amazonaws.com"
        }
    }
}
```

Your EMR Studio service role must also have the access permissions to use your AWS KMS key. The following is a sample IAM policy granting the minimum access permissions for EMR Studio Workspace storage encryption:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowEMRStudioWorkspaceStorageEncryptionAccess",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey",
        "kms:ReEncryptFrom",
        "kms:ReEncryptTo",
        "kms:DescribeKey"
      ],
      "Resource": [
        "arn:aws:kms:*:123456789012:key/12345678-1234-1234-1234-123456789012"
      ]
    }
  ]
}
```

Create a new EMR Studio

Follow these steps to create a new EMR Studio that uses workspace storage encryption.

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

2. Choose **Studios**, then choose **Create Studio**.
3. For **S3 location for storage**, enter or choose an Amazon S3 path. This is the Amazon S3 location where Amazon EMR stores workspace notebooks and files.
4. For **Service role**, enter or choose an IAM role. This is the IAM role that Amazon EMR assumes.
5. Choose **Encrypt Workspace files with your own AWS KMS key**.
6. Enter or choose an AWS KMS key to use to encrypt workspace notebooks and files in Amazon S3.
7. Choose **Create Studio** or **Create Studio and Launch Workspaces**.
8. Choose **Encrypt Workspace files with your own AWS KMS key**.
9. Enter or choose an AWS KMS to use to encrypt workspace notebooks and files in Amazon S3.
10. Choose **Save Changes**.

The following steps demonstrate how to update an EMR Studio and set up workspace storage encryption.

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose **an existing EMR Studio from the list**, then choose **Edit**.
3. Choose **Encrypt Workspace files with your own AWS KMS key**.
4. Enter or choose an AWS KMS to use to encrypt workspace notebooks and files in Amazon S3.
5. Choose **Save Changes**.

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 Amazon 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 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 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.

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 Service Catalog blog post.

Optional template parameters

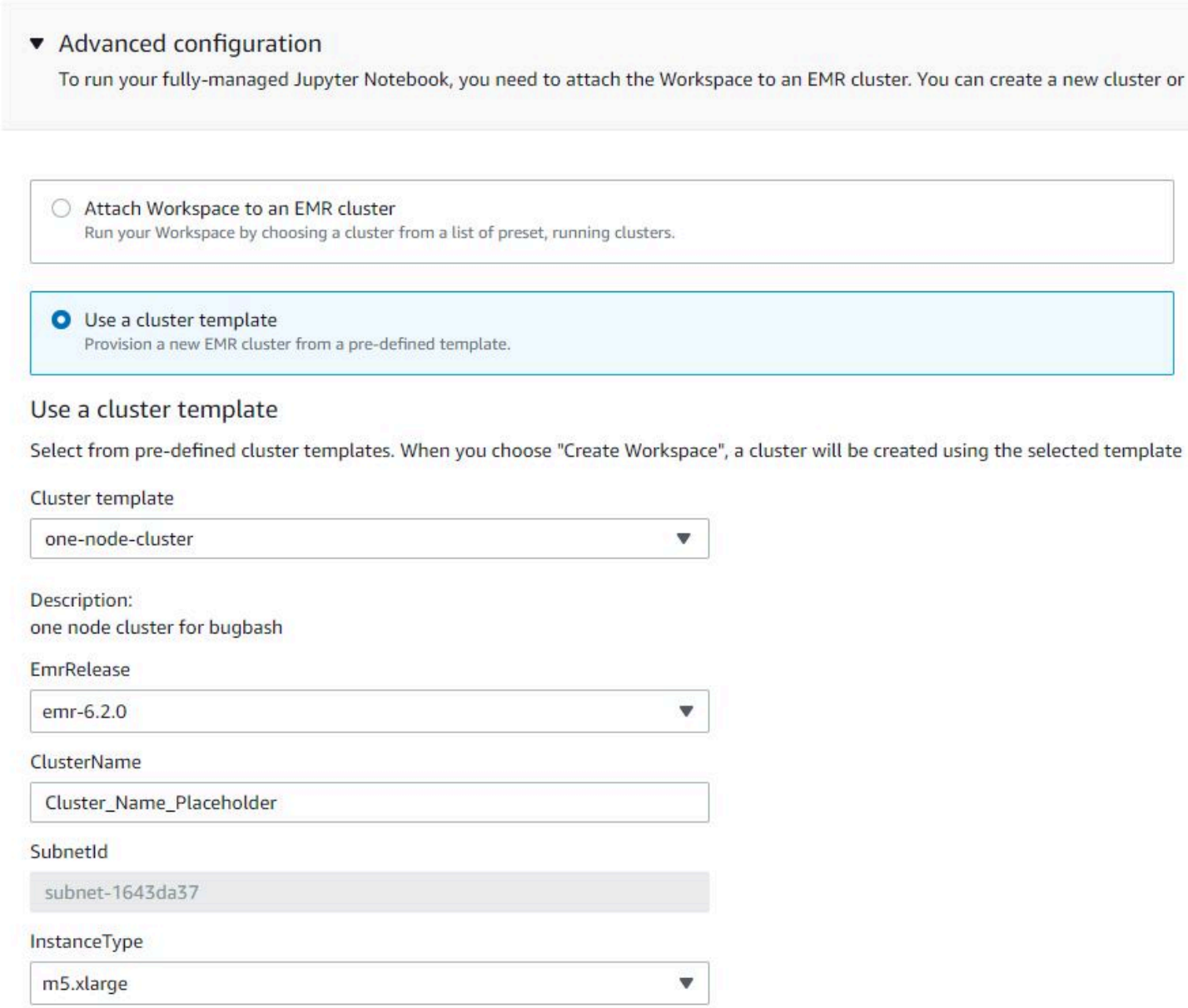
You can include additional options in the `Parameters` section of your template. *Parameters* let Studio users input or select custom values for a cluster. For example, you could add a parameter that lets users select a particular Amazon EMR release. For more information, see Parameters in the *AWS CloudFormation 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 Service Catalog administrator console view. You also need the required IAM permissions to perform Service

Catalog administrative tasks. For more information, see Grant permissions to Service Catalog administrators.

Create EMR cluster templates

To create EMR cluster templates using 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_V2
      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 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 a users
IAM federation	Your IAM role for federation
IAM Identity Center federation	Your EMR Studio user role

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.

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 when you launch a cluster.

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.

> **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 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.

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.

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 default Amazon S3 storage location in a folder called `life-cycle-configuration`. For example, if your default S3 location is `s3://amzn-s3-demo-bucket/workspace`, your configuration file would be in `s3://amzn-s3-demo-bucket/workspace/`**`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 primary node, improving resource utilization on the primary 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

This section contains topics that help you configure and interact with an Amazon EMR Studio.

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

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

- Learn EMR Studio workspaces
- Configure Workspace collaboration in EMR Studio
- Run an EMR Studio Workspace with a runtime role
- Run Amazon EMR Studio Workspace Workspace notebooks programmatically
- Browse data with SQL Explorer for EMR Studio
- Attach a compute to an EMR Studio Workspace
- Link Git-based repositories to an EMR Studio Workspace
- Use the Amazon Athena SQL editor in EMR Studio
- Amazon CodeWhisperer integration with EMR Studio Workspaces
- Debug applications and jobs with EMR Studio
- Install kernels and libraries in an EMR Studio Workspace
- Enhance kernels with magic commands in EMR Studio
- Use multi-language notebooks with Spark kernels

Learn EMR Studio workspaces

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 on the How Amazon EMR Studio works page.

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

- Create an EMR Studio Workspace
- Launch a Workspace in EMR Studio
- Understand the Workspace user interface in EMR Studio
- Explore notebook examples in an EMR Studio workspace

- Save Workspace content in EMR Studio
- Delete a Workspace and notebook files in EMR Studio
- Understand Workspace status
- Resolve Workspace connectivity issues

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 a Workspace**.
3. Enter a **Workspace name** and a **Description**. Naming a Workspace helps you identify it on the **Workspaces** page.
4. If you want to work with other Studio users in this Workspace in real time, enable Workspace collaboration. You can configure collaborators after you launch the Workspace.
5. If you want to attach a cluster to a Workspace, expand the **Advanced configuration** section. You can attach a cluster later, if you prefer. For more information, see Attach a compute to an EMR Studio Workspace.

   > **Note**
   >
   > To provision a new cluster, you need 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 and attach a new EMR cluster to an EMR Studio Workspace.
6. Choose **Create a 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 in EMR Studio

Launch a Workspace in EMR Studio

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**.

> **Note**
>
> If you had EMR notebooks in the old Amazon EMR console, you can find them in the console as EMR Studio Workspaces. EMR Notebooks users need additional IAM role permissions to access or create Workspaces. If you recently created a notebook in the old console, you might need to refresh the Workspaces list to see it in the console. For more information about the transition, see Amazon EMR Notebooks are available as Amazon EMR Studio Workspaces in the console and Managing Amazon EMR clusters with the console

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**. Alternatively, select the row for the Workspace and then select **Launch Workspace**. You can choose from the following launch options:
   - **Quick launch** – Quickly launch your Workspace with default options. Choose **Quick launch** if you want to attach clusters to the Workspace in JupyterLab.
   - **Launch with options** – Launch your Workspace with custom options. You can choose to launch in either Jupyter or JupyterLab, attach your Workspace to an EMR cluster, and select your security groups.

> **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 in EMR Studio

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 and attach a new EMR cluster to an EMR Studio Workspace.
- **Amazon 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.
- **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.

Explore notebook examples in an EMR Studio workspace

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 in EMR Studio

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.

Delete a Workspace and notebook files in EMR Studio

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*, its notebook files and folders will remain in the Amazon S3 storage location. The files continue to accrue storage charges. To avoid storage charges, remove all backed-up files and folders that are associated with your 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. EMR Studio removes the file or folder from Amazon S3 for you.

From the Workspace UI

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. If you want to remove the notebook files that were associated with the deleted Workspace from Amazon S3, follow the instructions for Deleting objects in the *Amazon Simple Storage Service Console User Guide*. If you did not create the Studio, consult your Studio administrator to determine the Amazon S3 backup location for the deleted Workspace.

From the Workspaces list

Delete a Workspace and its associated backup files from the Workspaces list

1. Navigate to the **Workspace**s list in the console.
2. Select the Workspace that you want to delete from the list and then choose **Actions**.

3. Choose **Delete**.
4. If you want to remove the notebook files that were associated with the deleted Workspace from Amazon S3, follow the instructions for Deleting objects in the *Amazon Simple Storage Service Console User Guide*. 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 has stopped. 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 shutting down and will be set to **Idle**. When you stop a Workspace

Status	Description
	, it terminates any corresponding notebook kernels. EMR Studio stops notebooks that have been inactive for a long time.
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. When you delete a Workspace, its notebook files will remain in the Amazon S3 storage location.

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 console.
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 in EMR Studio

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.

> **Note**
>
> EMR Studio Workspace collaboration isn't supported with EMR Serverless interactive applications or if trusted identity propagation is enabled.

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.

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.

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, a user can only add other a 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	a 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. 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 an EMR Studio Workspace with a runtime role

> **Note**
>
> The runtime role functionality described on this page only applies to Amazon EMR running on Amazon EC2, and doesn't refer to the runtime role functionality in EMR Serverless

interactive applications. To learn more about how to use runtime roles in EMR Serverless, see Job runtime roles in the *Amazon EMR Serverless User Guide*.

A *runtime role* is an AWS Identity and Access Management (IAM) role that you can specify when you submit a job or query to an Amazon EMR cluster. The job or query that you submit to your EMR cluster uses the runtime role to access AWS resources, such as objects in Amazon S3.

When you attach an EMR Studio Workspace to an EMR cluster that uses Amazon EMR 6.11 or higher, you can select a runtime role for the job or query that you submit to use when it accesses AWS resources. However, if the EMR cluster doesn't support runtime roles, the EMR cluster won't assume the role when it accesses AWS resources.

Before you can use a runtime role with an Amazon EMR Studio Workspace, an administrator must configure user permissions so that the Studio user can call the `elasticmapreduce:GetClusterSessionCredentials` API on the runtime role. Then, launch a new cluster with a runtime role that you can use with your Amazon EMR Studio Workspace.

On this page

- Configure user permissions for the runtime role
- Launch a new cluster with a runtime role
- Use the EMR cluster with a runtime role in Workspaces
- Considerations

Configure user permissions for the runtime role

Configure user permissions so that the Studio user can call the `elasticmapreduce:GetClusterSessionCredentials` API on the runtime role that the user wants to use. You must also configure the section called "Studio user permissions (EC2, EKS)" before the user can start using Studio.

> **Warning**
>
> To grant this permission, create a condition based on the `elasticmapreduce:ExecutionRoleArn` context key when you grant a caller access to call the `GetClusterSessionCredentials` APIs. The following examples demonstrate how to do so.

```
{
      "Sid": "AllowSpecificExecRoleArn",
      "Effect": "Allow",
      "Action": [
          "elasticmapreduce:GetClusterSessionCredentials"
      ],
      "Resource": "*",
      "Condition": {
          "StringEquals": {
              "elasticmapreduce:ExecutionRoleArn": [
                  "arn:aws:iam::111122223333:role/test-emr-demo1",
                  "arn:aws:iam::111122223333:role/test-emr-demo2"
              ]
          }
      }
  }
```

The following example demonstrates how to allow an IAM principal to use an IAM role named `test-emr-demo3` as the runtime role. Additionally, the policy holder will only be able to access Amazon EMR clusters with the cluster ID `j-123456789`.

```
{
    "Sid":"AllowSpecificExecRoleArn",
    "Effect":"Allow",
    "Action":[
        "elasticmapreduce:GetClusterSessionCredentials"
    ],
    "Resource": [
          "arn:aws:elasticmapreduce:<region>:111122223333:cluster/j-123456789"
     ],
    "Condition":{
        "StringEquals":{
            "elasticmapreduce:ExecutionRoleArn":[
                "arn:aws:iam::111122223333:role/test-emr-demo3"
            ]
        }
    }
}
```

The following example lets an IAM principal use any IAM role with a name starting with the string `test-emr-demo4` as the runtime role. Additionally, the policy holder will only be able to access Amazon EMR clusters tagged with the key-value pair `tagKey: tagValue`.

```
{
    "Sid":"AllowSpecificExecRoleArn",
    "Effect":"Allow",
    "Action":[
        "elasticmapreduce:GetClusterSessionCredentials"
    ],
    "Resource": "*",
    "Condition":{
        "StringEquals":{
             "elasticmapreduce:ResourceTag/tagKey": "tagValue"
        },
        "StringLike":{
            "elasticmapreduce:ExecutionRoleArn":[
                "arn:aws:iam::111122223333:role/test-emr-demo4*"
            ]
        }
    }
}
```

Launch a new cluster with a runtime role

Now that you have the required permissions, launch a new cluster with a runtime role that you can use with your Amazon EMR Studio Workspace.

If you have already launched a new cluster with a runtime role, you can skip to the the section called “Use the cluster with your Workspace” section.

1. First, complete the prerequisites in the Runtime roles for Amazon EMR steps section.
2. Then, launch a cluster with the following settings to use runtime roles with Amazon EMR Studio Workspaces. For instructions on how to launch your cluster, see Specify a security configuration for an Amazon EMR cluster.
   - Choose release label emr-6.11.0 or later.
   - Select Spark, Livy, and Jupyter Enterprise Gateway as your cluster applications.
   - Use the security configuration that you created in the previous step.
   - Optionally, you can enable Lake Formation for your EMR cluster. For more information, see Enable Lake Formation with Amazon EMR.

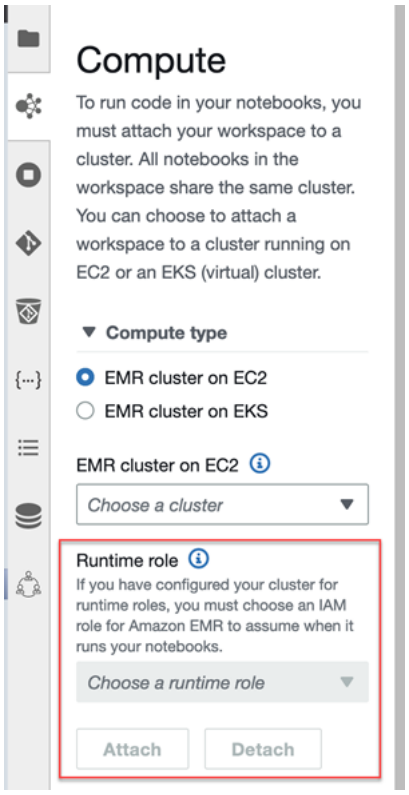
After you launch your cluster, you're ready to use the runtime role-enabled cluster with an EMR Studio Workspace.

> **Note**
>
> The ExecutionRoleArn value is currently not supported with the StartNotebookExecution API operation when the `ExecutionEngineConfig.Type` value is EMR.

Use the EMR cluster with a runtime role in Workspaces

Once you have set up and launched your cluster, you can use the runtime role-enabled cluster with your EMR Studio Workspace.

1. Create a new workspace or launch an existing workspace. For more information, see Create an EMR Studio Workspace.
2. Choose the **EMR clusters** tab in the left sidebar of your open Workspace, expand the **Compute type** section, and choose your cluster from the **EMR cluster on EC2** menu, and the runtime role from the **Runtime role** menu.



3. Choose **Attach** to attach the cluster with runtime role to your Workspace.

> **Note**
>
> When you choose a runtime role, note that it can have underlying managed policies associated with it. In most cases we recommend choosing limited resources, such as specific notebooks. If you choose a runtime role that includes access for all of your notebooks, for instance, the managed policy associated with the role provides full access.

Considerations

Keep in mind the following considerations when you use a runtime role-enabled cluster with your Amazon EMR Studio Workspace:

- You can only select a runtime role when you attach an EMR Studio Workspace to an EMR cluster that uses Amazon EMR release 6.11 or higher.
- The runtime role functionality described on this page is only supported with Amazon EMR running on Amazon EC2, and isn't supported with EMR Serverless interactive applications. To learn more about runtime roles for EMR Serverless, see Job runtime roles in the *Amazon EMR Serverless User Guide*.
- Although you need to configure additional permissions before you can specify a runtime role when submitting a job to a cluster, you don't need additional permissions to access the files generated by an EMR Studio Workspace. The permissions for such files are the same as files generated from clusters without runtime roles.
- You can't use SQL Explorer in an EMR Studio Workspace with a cluster that has a runtime role. Amazon EMR disables SQL Explorer in the UI when a Workspace is attached to a runtime role-enabled EMR cluster.
- You can't use collaboration mode in an EMR Studio Workspace with a cluster that has a runtime role. Amazon EMR disables Workspace collaboration capabilities when a Workspace is attached to a runtime role-enabled EMR cluster. The Workspace will remain accessible only to the user who attached the Workspace.
- You can't use runtime roles in a Studio with IAM Identity Center trusted identity propagation enabled.
- You might encounter a warning **"Page may not be safe!"** from Spark UI for a runtime role-enabled cluster that uses Amazon EMR release 7.4.0 and lower. If this happens, bypass the alert to continue to see the Spark UI.

Run Amazon EMR Studio Workspace Workspace notebooks programmatically

> **Note**
>
> Programmatic execution of notebooks isn't supported with Amazon EMR Serverless interactive applications.

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 programmatically, see Sample programmatic commands for EMR Notebooks.

Browse data with SQL Explorer for EMR Studio

> **Note**
>
> SQL Explorer for EMR Studio isn't supported with Amazon EMR Serverless interactive applications or in a Studio with IAM Identity Center trusted identity propagation enabled.

This topic provides information to help you get started with SQL Explorer in Amazon EMR Studio. 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 compute to an EMR Studio Workspace.
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 compute to an EMR Studio Workspace.
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 compute to an EMR Studio 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, to an Amazon EMR on EKS cluster, or to an EMR Serverless application. 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 an Amazon EC2 cluster to an EMR Studio Workspace
- Attach an Amazon EMR on EKS cluster to an EMR Studio Workspace
- Attach an Amazon EMR Serverless application to an EMR Studio Workspace
- Create and attach a new EMR cluster to an EMR Studio Workspace
- Detach a compute from an EMR Studio Workspace

Attach an Amazon EC2 cluster to an EMR Studio Workspace

You can attach an EMR cluster running on Amazon EC2 to a Workspace when you create the Workspace, or attach a cluster to an existing Workspace. If you want to create and attach a *new* cluster, see Create and attach a new EMR cluster to an EMR Studio Workspace.

> **Note**
>
> A workspace in a Studio that has IAM Identity Center trusted identity propagation enabled can only attach to an EMR cluster with a security configuration that has Identity Center enabled.

On create

Attach to an Amazon EMR compute 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 to the Workspace.

After you attach a cluster, finish creating the Workspace. When you open the new Workspace for the first time and choose the **EMR clusters** panel, you should see your selected cluster attached.

On launch

Attach to an Amazon EMR compute cluster when you launch the Workspace

1. Navigate to the Workspaces list and select the row for the Workspace that you want to launch. Then, select **Launch Workspace** > **Launch with options**.
2. Choose an EMR cluster to attach to your Workspace.

After you attach a cluster, finish creating the Workspace. When you open the new Workspace for the first time and choose the **EMR clusters** panel, you should see your selected cluster attached.

In JupyterLab

Attach a Workspace to an Amazon EMR compute cluster in JupyterLab

1. Select your Workspace, then select **Launch Workspace** > **Quick launch**.
2. Inside JupyterLab, open the **Cluster**tab in the left sidebar.
3. Select the **EMR on EC2 cluster** dropdown, or select an Amazon EMR on EKS cluster.
4. Select **Attach** to attach the cluster to your Workspace.

After you attach the cluster, finish creating the Workspace. When you open the new Workspace for the first time and choose the **EMR clusters** panel, you should see your selected cluster attached.

In the Workspace UI

Attach a Workspace to an Amazon EMR compute 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.

Attach an Amazon EMR on EKS cluster to an EMR Studio Workspace

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 Amazon EMR on EKS, see What is Amazon EMR on EKS.

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

> **Note**
>
> You can't launch an Amazon EMR on EKS cluster in a EMR Studio that uses IAM Identity Center trusted identity propagation.

On create

To attach an Amazon 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 Amazon EMR on EKS cluster**.
3. Under **Amazon 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 a 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.

In the Workspace UI

To attach an Amazon 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.

Attach an Amazon EMR Serverless application to an EMR Studio Workspace

You can attach a Workspace to an EMR Serverless application to run interactive workloads. For more information, see Using notebooks to run interactive workloads with EMR Serverless through EMR Studio.

> **Note**
>
> You can't attach an EMR Serverless application to a EMR Studio that uses IAM Identity Center trusted identity propagation.

Example Attach a Workspace to an EMR Serverless application in JupyterLab

Before you can connect a Workspace to an EMR Serverless application, your account administrator must grant you access permissions as described in Required permissions for interactive workloads.

1. Navigate to EMR Studio select your Workspace, then select **Launch Workspace** > **Quick launch**.
2. Inside JupyterLab, open the **Cluster** tab in the left sidebar.
3. Select **EMR Serverless** as a compute option, then select an EMR Serverless application and a runtime role.
4. To attach the cluster to your Workspace, choose **Attach**.

Now when you open this Workspace, you should see your selected application attached.

Create and attach a new EMR cluster to an EMR Studio Workspace

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.

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.

> **Note**
>
> If you enabled trusted identity propagation with IAM Identity Center for your Studio, then you must use a template to create a cluster.

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 **Amazon EMR release**, Choose an Amazon 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 instance types for use with Amazon EMR. One instance will be used as the primary 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 a 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 a 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 compute from an EMR Studio Workspace

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.

To detach an EMR Serverless application from an EMR Studio Workspace

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

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

Link Git-based repositories to an EMR Studio Workspace

Associate up to three Git-based repositories with an Amazon EMR Studio Workspace to save and share notebook files.

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 primary node of a cluster.

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

Alternatively, you can use the 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.
- 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 **Amazon 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 **Amazon 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–

Option	Description
	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 informati on, 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**.

Use the Amazon Athena SQL editor in EMR Studio

Overview

You can use Amazon EMR Studio to develop and run interactive queries on Amazon Athena. That means that you can perform SQL analytics on Athena from the same EMR Studio interface that you use to run your Spark, Scala, and other workloads. With this integration, you can use auto-completion to develop queries quickly, browse data in your AWS Glue Data Catalog, create saved queries, view your query history, and more.

For more information on using Amazon Athena, see Using Athena SQL in the *Amazon Athena User Guide*.

Use the Athena SQL editor in EMR Studio

Use the following steps to develop and run interactive queries on Amazon Athena from your EMR Studio:

1. Add the required permissions to the user role for the users who access the Workspaces in this Studio. The permissions are listed in the AWS Identity and Access Management permissions for EMR Studio users table in the column **Access Amazon Athena SQL editor from your EMR Studio**. Alternatively, you can choose to copy the **Advanced** policy contents from the Example user policies to grant users full permissions to EMR Studio capabilities including this one.
2. Set up and create an EMR Studio.
3. Navigate to your Studio and select **Query editor** from the sidebar.

You should now see the familiar Athena editor UI. For information on getting started and using Athena SQL to run interactive queries, see Getting started and Using Athena SQL in the *Amazon Athena User Guide*.

> **Note**
>
> If you have enabled trusted identity propagation through IAM Identity Center for your EMR Studio, then you must use Athena workgroups to control query access, and the workgroup that you use must also use trusted identity propagation. For steps to set up Identity Center and enable trusted identity propagation for your workgroup, see Using IAM Identity Center enabled Athena workgroups in the *Amazon Athena User Guide*.

Considerations for using the Athena SQL editor in EMR Studio

- Integration with Athena is available in all commercial Regions where EMR Studio and Athena are available.
- The following Athena features are not available in EMR Studio:
  - Admin features like creating or updating Athena workgroups, data sources, or capacity reservations
  - Athena for Spark or Spark notebooks
  - Amazon DataZone integration
  - Cost Based Optimizer (CBO)
  - Step functions

Amazon CodeWhisperer integration with EMR Studio Workspaces

Overview

You can use Amazon CodeWhisperer with Amazon EMR Studio to get real-time recommendations as you write code in JupyterLab. CodeWhisperer can complete your comments, finish single lines of code, make line-by-line recommendations, and generate fully-formed functions.

> **Note**
>
> When you use Amazon EMR Studio, AWS might store data about your usage and content for service improvement purposes. For more information and instructions to opt out of data sharing, see Sharing your data with AWS in the *Amazon CodeWhisperer User Guide*.

Considerations for using CodeWhisperer with Workspaces

- CodeWhisperer integration is available in the same AWS Regions where EMR Studio is available, as documented in the EMR Studio considerations.
- Amazon EMR Studio automatically uses the CodeWhisperer endpoint in US East (N. Virginia) (us-east-1) for recommendations, regardless of the Region that your studio is in.
- CodeWhisperer supports only Python language for coding ETL scripts for Spark jobs in EMR Studio.

- A client-side telemetry option quantifies your usage of CodeWhisperer. This functionality isn't supported with EMR Studio.

Permissions required for CodeWhisperer

To use CodeWhisperer, you must attach the following policy to your IAM user role for Amazon EMR Studio:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeWhispererPermissions",
      "Effect": "Allow",
      "Action": [
        "codewhisperer:GenerateRecommendations"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Use CodeWhisperer with Workspaces

To display the CodeWhisperer reference log in JupyterLab, open the **CodeWhisperer** panel at the bottom of the JupyterLab window and choose **Open Code Reference Log**.

The following list contains shortcuts that you can use to interact with CodeWhisperer suggestions:

- **Pause recommendations** – Use **Pause Auto-Suggestions** from the CodeWhisperer settings.
- **Accept a recommendation** – Press **Tab** on your keyboard.
- **Reject a recommendation** – Press **Escape** on your keyboard.
- **Navigate recommendations** – Use the **Up** and **Down** arrows on your keyboard.
- **Manual invoke** – Press **Alt** and **C** on your keyboard. If you're using a Mac, press **Cmd** and **C**.

You can also use CodeWhisperer to change settings like log level and get suggestions for code references. For more information, see Setting up CodeWhisperer with JupyterLab and Features in the *Amazon CodeWhisperer User Guide*.

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 in Amazon EMR.

> **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 compute to an EMR Studio Workspace.
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.

```
[1]: sc
```

```
Starting Spark application
```

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
2	application_1613085840432_0003	spark	idle	Link	Link	✔

```
SparkSession available as 'spark'.

res1: org.apache.spark.SparkContext = org.apache.spark.SparkContext@58262802
```

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 **Amazon EMR on EC2** on the left side of the page to open the **Amazon 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 EMR Studio running on EMR Serverless

Similar to Amazon EMR running on Amazon EC2, you can use the Workspace user interface to analyze your EMR Serverless applications. From the Workspace UI, when you use Amazon EMR releases 6.14.0 and higher, you can launch the Spark web user interface (the Spark UI or Spark

History Server) from a notebook in your Workspace. For your convenience, we also provide a link to the driver log for quick access the Spark driver logs.

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 **Amazon 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.

Kernels and libraries on clusters that run on Amazon EC2

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 primary 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 primary 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 in EMR Studio.

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

Kernels and libraries on EMR Serverless applications

Each EMR Serverless application comes with the following Python and PySpark libraries installed:

- **Python** – ggplot, matplotlib, numpy, pandas, plotly, bokeh, scikit-learn, scipy, seaborn
- **PySpark** – ggplot, matplotlib,numpy, pandas, plotly, bokeh, scikit-learn, scipy, seaborn

Enhance kernels with magic commands in EMR Studio

Overview

EMR Studio and EMR Notebooks support magic commands. *Magic* commands, or *magics*, are enhancements that the IPython kernel provides to help you run and analyze data. IPython is an interactive shell environment that is 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

For all available magics, see List magic and Sparkmagic commands.

Considerations and limitations

- EMR Serverless doesn't support `%%sh` to run `spark-submit`. It doesn't support the EMR Notebooks magics.
- Amazon EMR on EKS clusters don't support Sparkmagic commands for EMR Studio. This is because Spark kernels that you use with managed endpoints are built into Kubernetes, and they aren't 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')
```

- The following magic commands and actions are prohibited by AWS:
  - `%alias`
  - `%alias_magic`
  - `%automagic`
  - `%macro`
  - Modifying `proxy_user` with `%configure`
  - Modifying `KERNEL_USERNAME` with `%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.

Use `%%configure` to configure Spark

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 that's mentioned in the configuration documentation for Apache Spark.

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://amzn-s3-demo-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"
     }
}
```

Use `%%sh` to run `spark-submit`

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://amzn-s3-demo-bucket/test.py
```

In this example, the cluster needs access to the location of `s3://amzn-s3-demo-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 for more information.

Use `%%display` to visualize Spark dataframes

You can use the `%%display` magic to visualize a Spark dataframe. To use this magic, run the following command.

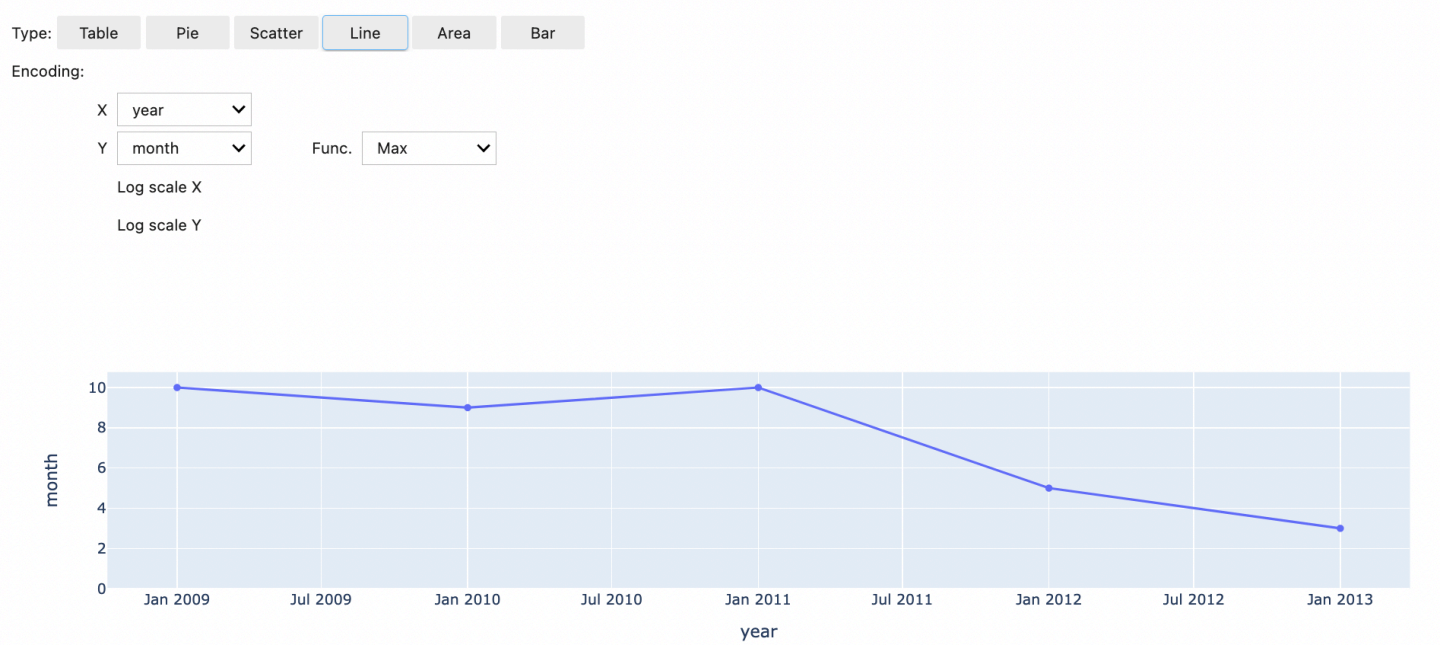
```
%%display df
```

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) 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 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.

  The Amazon EMR 7.x series uses Amazon Linux 2023, which doesn't support EPEL repositories. If you're running Amazon EMR 7.x, follow the s3fs-fuse GitHub instructions to install `s3fs-fuse`. If you use the 5.x or 6.x series, use the following commands to install `s3fs-fuse`.

```
#!/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

> **Note**
>
> With Amazon EMR releases 6.0 through 6.9.0, and 5.0 through 5.36.0, only `emr-notebooks-magics` package versions 0.2.0 and higher support `%mount_workspace_dir` magic.

Complete the following steps to install EMR Notebooks magics.

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

   ```
   %pip install boto3 --upgrade
   %pip install botocore --upgrade
   %pip install emr-notebooks-magics --upgrade
   ```

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?
```

Run a notebook in headless mode with `%execute_notebook`

With `%execute_notebook` magic, you can run another notebook in headless mode and view the output for each cell that you've run. This magic requires additional permissions for the instance role that Amazon EMR and Amazon EC2 share. For more details on how to grant additional permissions, run the command `%execute_notebook?`.

During a long-running job, your system might go to sleep because of inactivity, or might temporarily lose internet connectivity. This might disrupt the connection between your browser and the Jupyter Server. In this case, you might lose the output from the cells that you've run and sent from the Jupyter Server.

If you run the notebook in headless mode with `%execute_notebook` magic, EMR Notebooks captures output from the cells that have run, even if the local network experiences disruption. EMR Notebooks saves the output incrementally in a new notebook with the same name as the notebook that you've run. EMR Notebooks then places the notebook into a new folder within the workspace. Headless runs occur on the same cluster and uses service role `EMR_Notebook_DefaultRole`, but additional arguments can alter the default values.

To run a notebook in headless mode, use the following command:

```
%execute_notebook <relative-file-path>
```

To specify a cluster ID and service role for a headless run, use the following command:

```
%execute_notebook <notebook_name>.ipynb --cluster-id <emr-cluster-id> --service-role
 <emr-notebook-service-role>
```

When Amazon EMR and Amazon EC2 share an instance role, the role requires the following additional permissions:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:StartNotebookExecution",
```

```
        "elasticmapreduce:DescribeNotebookExecution",
        "ec2:DescribeInstances"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowELASTICMAPREDUCEStartnotebookexecution"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::123456789012:role/EMR_Notebooks_DefaultRole"
      ],
      "Sid": "AllowIAMPassrole"
    }
  ]
}
```

Note

To use `%execute_notebook` magic, install the `emr-notebooks-magics` package, version 0.2.3 or higher.

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`

Language	Command
Scala	`%%scalaspark`
R	`%%rspark` Not supported for interactive workloads with EMR Serverless.
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)
```

Amazon EMR Notebooks overview

> **Note**
>
> EMR Notebooks are available as EMR Studio Workspaces in the console. The **Create Workspace** button in the console lets you create new notebooks. To access or create Workspaces, EMR Notebooks users need additional IAM role permissions. For more information, see Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console and Amazon EMR console.

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 — 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 Amazon EMR API, without the need to interact with Amazon 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. Amazon EMR creates and saves the output notebook on S3 for each run of the parameterized notebook. For EMR notebook API code samples, see Sample programmatic commands for EMR Notebooks.

> **Important**
>
> The EMR Notebooks capability supports clusters that use Amazon EMR releases 5.18.0 and higher. We recommend that you use EMR Notebooks with clusters that use the latest

version of Amazon EMR, or at least 5.30.0, 5.32.0, or 6.2.0. With these releases, Jupyter kernels run on the attached cluster rather than on a Jupyter instance. This improves performance and enhances your ability to customize kernels and libraries. For more information, see Differences in capabilities by cluster release version.

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

Amazon EMR Notebooks are available as Amazon EMR Studio Workspaces in the console

Making the transition from EMR Notebooks to Workspaces

In the new Amazon EMR console, we've merged EMR Notebooks with Amazon EMR Studio Workspaces into a single experience. When you use an EMR Studio, you can create and configure different Workspaces to organize and run notebooks. If you had Amazon EMR notebooks in the old console, they're available as EMR Studio Workspaces in the console.

Amazon EMR created these new EMR Studio Workspaces for you. The number of Studios that we created corresponds to the number of distinct VPCs that you use from EMR Notebooks. For example, if you connect to EMR clusters in two different VPCs from EMR Notebooks, then we created two new EMR Studios. Your notebooks are distributed among the new Studios.

> **Important**
>
> We turned off the option to create new notebooks in the old Amazon EMR console. Instead, use **Create Workspace** in the new Amazon EMR console.

For more information on Amazon EMR Studio Workspaces, see Learn EMR Studio workspaces. For a conceptual overview of EMR Studio, see Workspaces on the How Amazon EMR Studio works page.

What do you need to do?

While you can still use your existing notebooks in the old console, we recommend that you instead use Amazon EMR Studio Workspaces in the console. You must configure additional role permissions to turn on the capabilities in EMR Studio that aren't available in EMR Notebooks.

> **Note**
>
> At a minimum, to view existing EMR Notebooks as EMR Studio Workspaces and to create new Workspaces, users must have `elasticmapreduce:ListStudios` and `elasticmapreduce:CreateStudioPresignedUrl` permissions on their roles. To access all of the EMR Studio features, see Enabling EMR Studio features for EMR Notebooks users for the complete list of added permissions that EMR Notebooks users will need.

Enhanced capabilities in EMR Studio beyond EMR Notebooks

With Amazon EMR Studio, you can set up and use the following capabilities that aren't available with EMR Notebooks:

- Browse and attach to EMR clusters from within Jupyterlab
- Browse and attach to EMR Notebooks virtual clusters from within Jupyterlab
- Connect to Git repos from within Jupyterlab
- Collaborate with other members of your team to write and run notebook code
- Browse data with SQL Explorer
- Provision EMR clusters with Service Catalog

For a complete list of capabilities with Amazon EMR Studio, see Key features of EMR Studio.

Enabling EMR Studio features for EMR Notebooks users

The new EMR Studios that we will create as part of this merge use the existing `EMR_Notebooks_DefaultRole` IAM role as the EMR Studio service role.

Users who transition to EMR Studio from EMR Notebooks and want to use the additional capabilities of EMR Studio require several new role permissions. Add the following permissions to the roles of your EMR Notebooks users who plan to use EMR Studio.

> **Note**
>
> At a minimum, to view existing EMR Notebooks as EMR Studio Workspaces and to create new Workspaces, users must have `elasticmapreduce:ListStudios` and

`elasticmapreduce:CreateStudioPresignedUrl` permissions on their roles. To use all of the EMR Studio features, add all of the permissions listed below. Admin users also need permission to create and manage an EMR Studio. For more information, see Administrator permissions to create and manage an EMR Studio.

```
"elasticmapreduce:DescribeStudio",
"elasticmapreduce:ListStudios",
"elasticmapreduce:CreateStudioPresignedUrl",
"elasticmapreduce:UpdateEditor",
"elasticmapreduce:PutWorkspaceAccess",
"elasticmapreduce:DeleteWorkspaceAccess",
"elasticmapreduce:ListWorkspaceAccessIdentities",
"emr-containers:ListVirtualClusters",
"emr-containers:DescribeVirtualCluster",
"emr-containers:ListManagedEndpoints",
"emr-containers:DescribeManagedEndpoint",
"emr-containers:CreateAccessTokenForManagedEndpoint",
"emr-containers:ListJobRuns",
"emr-containers:DescribeJobRun",
"servicecatalog:SearchProducts",
"servicecatalog:DescribeProduct",
"servicecatalog:DescribeProductView",
"servicecatalog:DescribeProvisioningParameters",
"servicecatalog:ProvisionProduct",
"servicecatalog:UpdateProvisionedProduct",
"servicecatalog:ListProvisioningArtifacts",
"servicecatalog:DescribeRecord",
"servicecatalog:ListLaunchPaths",
"cloudformation:DescribeStackResources"
```

The following permissions are also required to use the collaboration capabilities in EMR Studio, but weren't required with EMR Notebooks.

```
"sso-directory:SearchUsers",
"iam:GetUser",
"iam:GetRole",
"iam:ListUsers",
"iam:ListRoles",
"sso:GetManagedApplicationInstance"
```

Requirements, differences in release versions, and security for EMR Notebooks

> **Note**
>
> EMR Notebooks are available as EMR Studio Workspaces in the console. The **Create Workspace** button in the console lets you create new notebooks. To access or create Workspaces, EMR Notebooks users need additional IAM role permissions. For more information, see Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console and Amazon EMR console.

Consider the following requirements, differences in release versions, security information and other considerations 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 and Control network traffic with security groups for your Amazon EMR cluster.
- **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”.
  - 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 Amazon 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 primary 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 primary 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 primary node.
- **Ability to install Python libraries** – You can install Python libraries on the cluster primary node using `conda` and `pip`. We recommend using `conda`. With earlier versions, only notebook-scoped libraries 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 primary 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	No	No
5.30.0	Yes	Yes	Yes
6.0.0	No	No	No
5.32.0 and later, and 6.2.0 and later	Yes	Yes	Yes

Limits for concurrently attached EMR Notebooks

When you create a cluster that supports notebooks, consider the EC2 Instance type of the cluster primary 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.

Primary 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.

Security-related considerations

Using encrypted S3 locations

If you specify an encrypted location in Amazon S3 to store notebook files, you must set up the Service role for EMR Notebooks 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.

Using cookies with hosting domains

To augment the security for the off-console applications that you might use with Amazon EMR, the application hosting domains are registered in the Public Suffix List (PSL). Examples of these hosting domains include the following: `emrstudio-prod.us-east-1.amazonaws.com`, `emrnotebooks-prod.us-east-1.amazonaws.com`, `emrappui-prod.us-east-1.amazonaws.com`. For further security, if you ever need to set sensitive cookies in the default domain name, we recommend that you use cookies with a `__Host-` prefix. This helps to defend your domain against cross-site request forgery attempts (CSRF). For more information, see the Set-Cookie page in the *Mozilla Developer Network*.

Create a Notebook in EMR Studio

> **Note**
>
> EMR Notebooks are available as EMR Studio Workspaces in the console. The **Create Workspace** button in the console lets you create new notebooks. To access or create Workspaces, EMR Notebooks users need additional IAM role permissions. For more information, see Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console and Amazon EMR console.

You create an EMR notebook using the old 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 Requirements, differences in release versions, and security for EMR Notebooks.

   —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 (5.36.2).
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 primary 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.
EMR role	Leave the default or choose the link to specify a custom service role for Amazon

Setting	Description
	EMR. For more information, see Service role for Amazon EMR (EMR role).
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).
EC2 key pair	Choose an EC2 key pair to be able to connect to cluster instances. For more information, see Connect to the Amazon EMR cluster primary node using SSH.
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 for Amazon EMR cluster cleanup.

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 primary instance and another for the notebook client instance. For more information, see the section called “Security groups for EMR Notebooks”.
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.
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://amzn-s3-demo-bucket/MyNotebooks` for a notebook named `MyFirstEMRManagedNotebook`, the notebook file is saved to `s3://amzn-s3-demo-bucket/MyNotebooks/NotebookID/MyFirstEMRManagedNotebook.ipynb`.

If you specify an encrypted location in Amazon S3, you must set up the Service role for EMR Notebooks 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.
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.

10. Choose **Create Notebook**.

Working with EMR Notebooks

> **Note**
>
> EMR Notebooks are available as EMR Studio Workspaces in the console. The **Create Workspace** button in the console lets you create new notebooks. To access or create Workspaces, EMR Notebooks users need additional IAM role permissions. For more information, see Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console and Amazon EMR console.

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

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

Understanding Notebook status

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

Status	Meaning
Ready	You can open the notebook using the notebook editor. While a notebook has a **Ready** status, you can stop or delete it. To change clusters, you must stop the notebook first. If a notebook in the **Ready** status is idle for a long period of time, it is stopped automatically.
Starting	The notebook is being created and attached to the cluster. While a notebook is starting, you cannot open the notebook editor, stop it, delete it, or change clusters.
Pending	The notebook has been created, and is waiting for integration with the cluster to complete. The cluster may still be provisioning resources or responding to other requests. You can open the notebook editor with the notebook in *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 terminati ng. While a notebook is stopping, you can't

Status	Meaning
	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 a 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" and Service role for EMR Notebooks.

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://amzn-s3-demo-bucket/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.
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 amzn-s3-demo-bucket/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://amzn-s3-demo-bucket/
MyNotebooksFolder/-12A3BCDEFJHIJKLMNO45PQRST/MyNotebook.ipynb
```

Sample programmatic commands for EMR Notebooks

Overview

You can execute EMR notebooks with execution APIs from a script or from command line. When you start, stop, list, and describe EMR notebook executions outside of the AWS console, you can programmatically control an EMR notebook. You can pass different parameter values to a notebook with a parameterized notebook cell. This eliminates the need to create a copy of the notebook for each new set of parameter values. For more information, see Amazon EMR API actions.

You can schedule or batch EMR notebook executions with Amazon CloudWatch events and AWS Lambda. For more information, see Using AWS Lambda with Amazon CloudWatch Events.

> **Note**
>
> EMR Notebooks are available as EMR Studio Workspaces in the console. The **Create Workspace** button in the console lets you create new notebooks. To access or create Workspaces, EMR Notebooks users need additional IAM role permissions. For more information, see Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console and Amazon EMR console.

Role permissions for programmatic execution

To use programmatic execution with EMR Notebooks, you must configure user permissions with the following policies:

JSON

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

```
      "Sid": "AllowExecutionActions",
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:StartNotebookExecution",
        "elasticmapreduce:DescribeNotebookExecution",
        "elasticmapreduce:ListNotebookExecutions"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "AllowPassingServiceRole",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::123456789012:role/EMR_Notebooks_DefaultRole"
      ]
    }
  ]
}
```

When you programmatically execute EMR Notebooks on an EMR Notebooks cluster, you must add these additional permissions:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowRetrievingManagedEndpointCredentials",
      "Effect": "Allow",
      "Action": [
        "emr-containers:GetManagedEndpointSessionCredentials"
      ],
      "Resource": [
        "arn:aws:emr-containers:*:123456789012:/virtualclusters/virtual-cluster-
id/endpoints/managed-endpoint-id"
```

```
      ],
      "Condition": {
        "StringEquals": {
          "emr-containers:ExecutionRoleArn": [
            "arn:aws:iam::123456789012:role/emr-on-eks-execution-role"
          ]
        }
      }
    },
    {
      "Sid": "AllowDescribingManagedEndpoint",
      "Effect": "Allow",
      "Action": [
        "emr-containers:DescribeManagedEndpoint"
      ],
      "Resource": [
        "arn:aws:emr-containers:*:123456789012:/virtualclusters/virtual-cluster-
id/endpoints/managed-endpoint-id"
      ]
    }
  ]
}
```

Limitations with programmatic execution

- A maximum of 100 concurrent executions are supported per AWS Region per account.
- An execution is terminated if it runs for more than 30 days.
- Programmatic execution of notebooks isn't supported with Amazon EMR Serverless interactive applications.

Examples of programmatic EMR notebook execution

The following sections provide several examples of programmatic EMR notebook execution with the AWS CLI, Boto3 SDK (Python), and Ruby:

- Notebook CLI command samples in EMR Studio
- Python samples for an EMR notebook
- Ruby samples for an EMR notebook

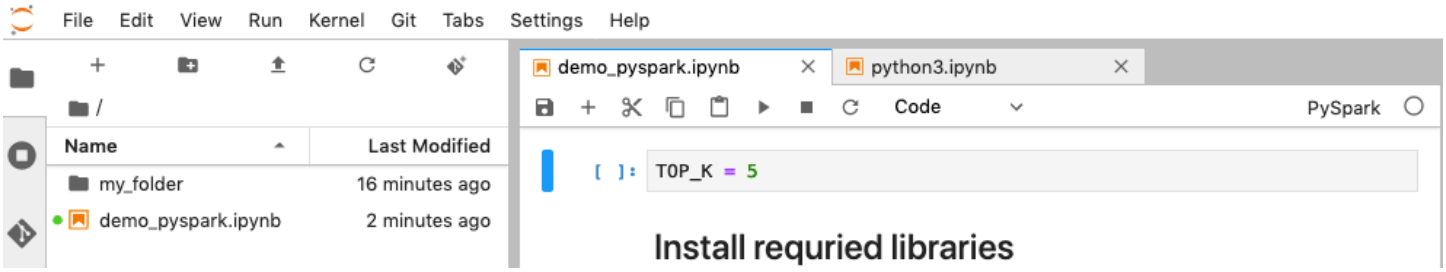
You can also run parameterized notebooks as part of scheduled workflows with an orchestration tool such as Apache Airflow or Amazon Managed Workflows for Apache Airflow (MWAA). For more information, see Orchestrating analytics jobs on EMR Notebooks using MWAA in the *AWS Big Data Blog*.

Notebook CLI command samples in EMR Studio

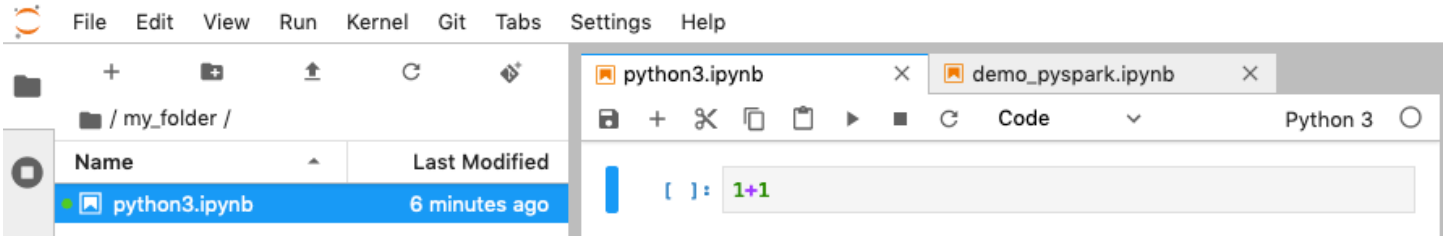
This topic shows CLI command samples for an EMR notebook. The example uses the demo notebook from the EMR Notebooks console. To locate the notebook, use the file path relative to the home directory. In this example, there are two notebook files that you can run: `demo_pyspark.ipynb` and `my_folder/python3.ipynb`.

> **Note**
>
> EMR Notebooks are available as EMR Studio Workspaces in the console. The **Create Workspace** button in the console lets you create new notebooks. To access or create Workspaces, EMR Notebooks users need additional IAM role permissions. For more information, see Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console and Amazon EMR console.

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 information about the Amazon EMR API `NotebookExecution` actions, see Amazon EMR API actions..

Run a notebook

You can use the AWS CLI to run your notebook with the `start-notebook-execution` action, as the following examples demonstrate.

Example – Executing an EMR notebook in an EMR Studio Workspace with an Amazon EMR (running on Amazon EC2) cluster

```
aws emr --region us-east-1 \
start-notebook-execution \
--editor-id e-ABCDEFG123456 \
--notebook-params '{"input_param":"my-value", "good_superhero":["superman", "batman"]}'
 \
--relative-path test.ipynb \
--notebook-execution-name my-execution \
--execution-engine '{"Id" : "j-1234ABCD123"}' \
--service-role EMR_Notebooks_DefaultRole

{
    "NotebookExecutionId": "ex-ABCDEFGHIJ1234ABCD"
}
```

Example – Executing an EMR notebook in an EMR Studio Workspace with an EMR Notebooks cluster

```
aws emr start-notebook-execution \
    --region us-east-1 \
    --service-role EMR_Notebooks_DefaultRole \
    --environment-variables '{"KERNEL_EXTRA_SPARK_OPTS": "--conf
 spark.executor.instances=1", "KERNEL_LAUNCH_TIMEOUT": "350"}' \
    --output-notebook-format HTML \
    --execution-engine Id=arn:aws:emr-containers:us-west-2:account-id:/
virtualclusters/ABCDEFG/
endpoints/ABCDEF,Type=EMR_ON_EKS,ExecutionRoleArn=arn:aws:iam::account-
id:role/execution-role \
    --editor-id e-ABCDEFG \
    --relative-path EMRonEKS-spark_python.ipynb
```

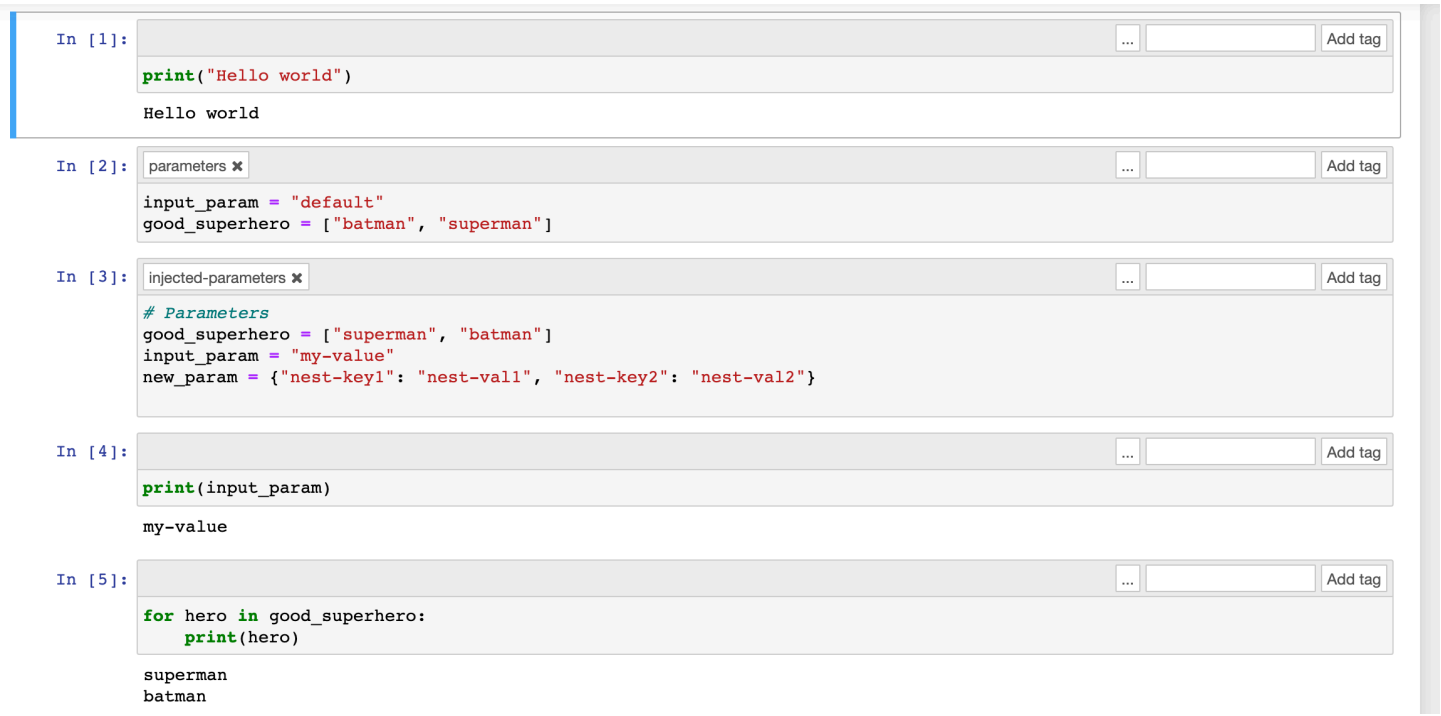
Example – Executing an EMR notebook specifying its Amazon S3 location

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

```
    --notebook-execution-name my-execution-on-emr-on-eks-cluster \
    --service-role EMR_Notebooks_DefaultRole \
    --environment-variables '{"KERNEL_EXTRA_SPARK_OPTS": "--conf
 spark.executor.instances=1", "KERNEL_LAUNCH_TIMEOUT": "350"}' \
    --output-notebook-format HTML \
    --execution-engine Id=arn:aws:emr-containers:us-west-2:account-id:/
virtualclusters/ABCDEF/
endpoints/ABCDEF,Type=EMR_ON_EKS,ExecutionRoleArn=arn:aws:iam::account-
id:role/execution-role \
    --notebook-s3-location '{"Bucket": "amzn-s3-demo-bucket","Key": "s3-prefix-to-
notebook-location/EMRonEKS-spark_python.ipynb"}' \
    --output-notebook-s3-location '{"Bucket": "amzn-s3-demo-bucket","Key": "s3-prefix-
for-storing-output-notebook"}'
```

Notebook output

Here's the output from a sample notebook. 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 the executions for a notebook 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 the executions for a notebook 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
        }
    ]
}
```

Python samples for an EMR notebook

This topic contains a sample command file. The code example is an SDK for Python (Boto3) file called `demo.py`. It shows the notebook execution APIs.

> **Note**
>
> EMR Notebooks are available as EMR Studio Workspaces in the console. The **Create Workspace** button in the console lets you create new notebooks. To access or create Workspaces, EMR Notebooks users need additional IAM role permissions. For more information, see Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console and Amazon EMR console.

For information about the Amazon EMR API `NotebookExecution` actions, see Amazon 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}}
```

Ruby samples for an EMR notebook

This topic contains a Ruby sample that demonstrate notebook functionality.

> **Note**
>
> EMR Notebooks are available as EMR Studio Workspaces in the console. The **Create Workspace** button in the console lets you create new notebooks. To access or create Workspaces, EMR Notebooks users need additional IAM role permissions. For more information, see Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console and Amazon EMR console.

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

```
# prepare an Amazon 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 Amazon S3 editor and EMR notebook are `s3://amzn-s3-demo-bucket/notebooks/e-EA8VGAA429FEQTC8HC9ZHWISK/test.ipynb`.

For information about the Amazon EMR API `NotebookExecution` actions, see Amazon 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=>n
: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 primary node.

> **Note**
>
> EMR Notebooks are available as EMR Studio Workspaces in the console. The **Create Workspace** button in the console lets you create new notebooks. To access or create Workspaces, EMR Notebooks users need additional IAM role permissions. For more information, see Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console and Amazon EMR console.

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 a user identity that ran the code instead. HDFS user directories on the primary 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 primary 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:

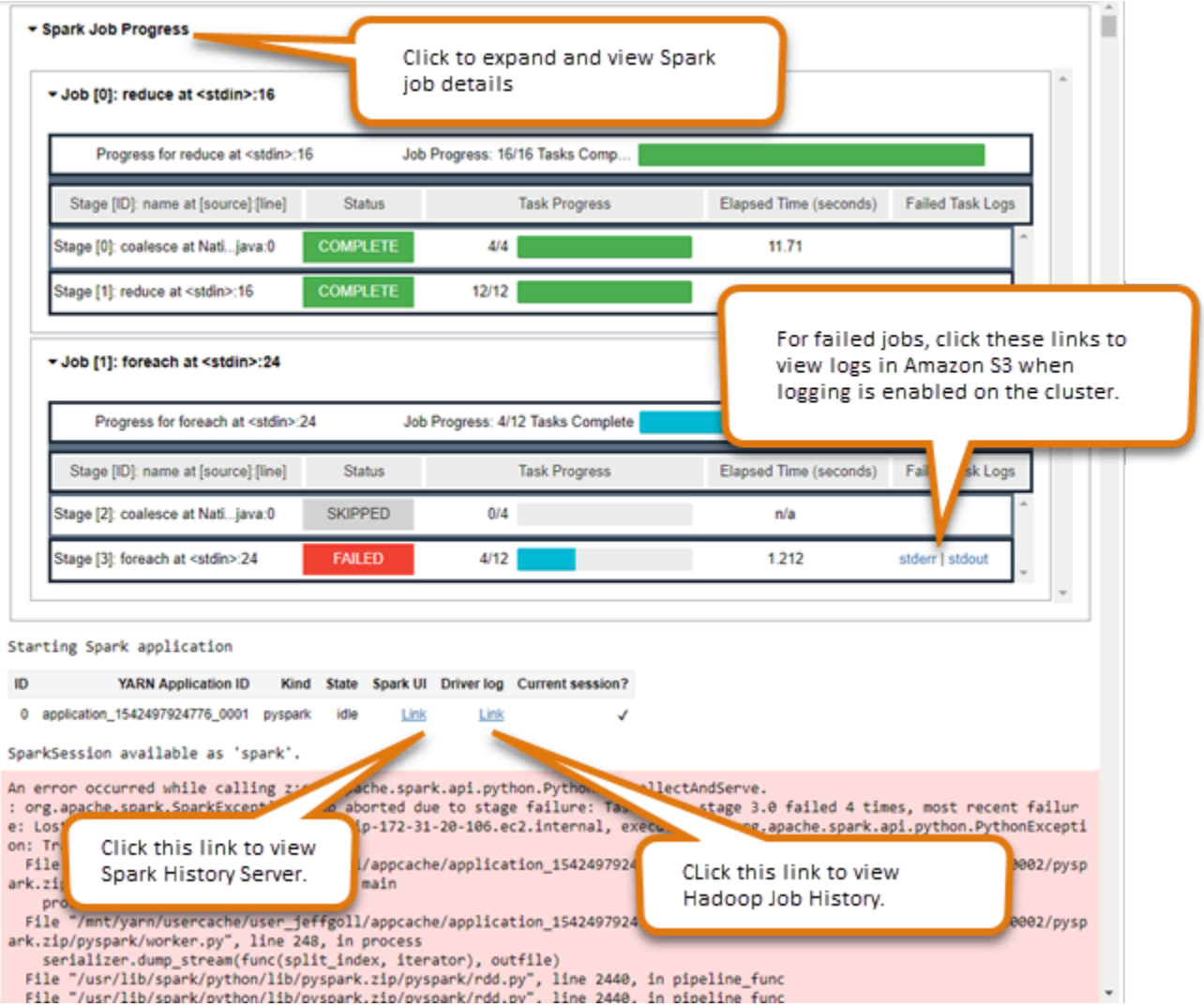
```
[
    {
        "Classification": "core-site",
        "Properties": {
          "hadoop.proxyuser.livy.groups": "*",
          "hadoop.proxyuser.livy.hosts": "*"
        }
    },
    {
        "Classification": "livy-conf",
        "Properties": {
          "livy.impersonation.enabled": "true"
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

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 primary node, you must set up an SSH client and proxy as appropriate. For more information, see View web interfaces hosted on Amazon EMR clusters. 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.

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.

- You can use AWS Identity and Access Management policy statements together with notebook tags to limit access. For more information, see How Amazon EMR works with IAM and Example identity-based policy statements for EMR Notebooks.

- Amazon EC2 security groups act as virtual firewalls that control network traffic between the cluster's primary 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.
- 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.

> **Note**
>
> EMR Notebooks are available as EMR Studio Workspaces in the console. The **Create Workspace** button in the console lets you create new notebooks. To access or create Workspaces, EMR Notebooks users need additional IAM role permissions. For more information, see Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console and Amazon EMR console.

Installing and using kernels and libraries in EMR Studio

Each EMR notebook comes with a set of pre-installed libraries and kernels. You can install additional libraries and kernels in an 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*.

> **Note**
>
> EMR Notebooks are available as EMR Studio Workspaces in the console. The **Create Workspace** button in the console lets you create new notebooks. To access or create Workspaces, EMR Notebooks users need additional IAM role permissions. For more information, see Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console and Amazon EMR console.

EMR Serverless applications come with the following pre-installed libraries for Python and PySpark:

- **Python libraries** – ggplot, matplotlib, numpy, pandas, plotly, bokeh, scikit-learn, scipy, scipy
- **PySpark libraries** – ggplot, matplotlib, numpy, pandas, plotly, bokeh, scikit-learn, scipy, scipy

Installing kernels and Python libraries on a cluster primary 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 primary 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 primary node. The libraries are not installed on core or task nodes and are not available to executors running on those nodes.

> **Note**
>
> For Amazon 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 primary 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 primary 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 primary 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://amzn-s3-demo-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://amzn-s3-demo-bucket/prefix/
my_pyspark_venv.tar.gz#environment",
   "spark.submit.deployMode":"cluster"
   }
}
```

You can also use `conda` to install Python libraries. You don't need sudo access to use `conda`. You must connect to the primary node with SSH, and then run `conda` from the terminal. For more information, see Connect to the Amazon EMR cluster primary node using SSH.

Example – Installing kernels

The following example demonstrates installing the Kotlin kernel using a terminal command while connected to the primary 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.

Considerations and limitations with notebook-scoped libraries

When you use notebook-scoped libraries, consider the following:

- Notebook-scoped libraries are available for clusters that you create with Amazon EMR releases 5.26.0 and higher.
- 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.
- You can uninstall only the libraries that were installed with the `install_pypi_package` API. You cannot uninstall any libraries that were 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

> **Note**
>
> EMR Notebooks are available as EMR Studio Workspaces in the console. The **Create Workspace** button in the console lets you create new notebooks. To access or create Workspaces, EMR Notebooks users need additional IAM role permissions. For more information, see Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console and Amazon EMR console.

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 when integrating an EMR notebook with a repository
- Add a Git-based repository to Amazon EMR
- Update or delete a Git-based repository from an EMR Studio Workspace
- Link or unlink a Git-based repository in EMR Studio
- Create a new Notebook with an associated Git repository in EMR Studio
- Use Git repositories in an EMR Studio Notebook

Prerequisites and considerations when integrating an EMR notebook with a repository

Consider the following best practices regarding commits, permissions, and hosting when planning to integrate a Git-based repository with EMR Notebooks.

> **Note**
>
> EMR Notebooks are available as EMR Studio Workspaces in the console. The **Create Workspace** button in the console lets you create new notebooks. To access or create Workspaces, EMR Notebooks users need additional IAM role permissions. For more

information, see Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console and Amazon EMR console.

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.

- **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" 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://amzn-s3-demo-bucket/notebooks`, your configuration file should be located at `s3://amzn-s3-demo-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

Refer to the following sections for information on how to add a Git-based repository to an EMR notebook in the old console, or to an EMR Studio Workspace in the console.

> **Note**
>
> EMR Notebooks are available as EMR Studio Workspaces in the console. The **Create Workspace** button in the console lets you create new notebooks. To access or create Workspaces, EMR Notebooks users need additional IAM role permissions. For more information, see Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console and Amazon EMR console.

Console

Because EMR Notebooks are EMR Studio Workspaces in the new console, you can follow the instructions in Link Git-based repositories to an EMR Studio Workspace to associate up to three Git repositories with your Workspace.

Alternatively, you can use the JupyterLab Git extension. Choose the **Git** icon from the left sidebar of your Jupyterlab notebook to access the extension. For information about the extension, see the jupyterlab-git GitHub repo.

To associate a Git repository with a Workspace, your Studio administrator must take steps to configure the Studio to allow Git repository linking. For more information, see Establish access and permissions for Git-based repositories.

Update or delete a Git-based repository from an EMR Studio Workspace

Refer to the following sections for information on how to delete a Git-based repository from an EMR notebook in the old console, or from an EMR Studio Workspace in the console.

> **Note**
>
> EMR Notebooks are available as EMR Studio Workspaces in the console. The **Create Workspace** button in the console lets you create new notebooks. To access or create Workspaces, EMR Notebooks users need additional IAM role permissions. For more

information, see Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console and Amazon EMR console.

Console

Because EMR Notebooks are EMR Studio Workspaces in the new console, you can you can refer to Link Git-based repositories to an EMR Studio Workspace for more information on working with Git repositories in your Workspace. But at this time, you can't delete Git repositories from Workspaces.

Link or unlink a Git-based repository in EMR Studio

Use the following steps to link or unlink a Git-based repository to an EMR notebook in the old console, or to an EMR Studio Workspace in the console.

> **Note**
>
> EMR Notebooks are available as EMR Studio Workspaces in the console. The **Create Workspace** button in the console lets you create new notebooks. To access or create Workspaces, EMR Notebooks users need additional IAM role permissions. For more information, see Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console and Amazon EMR console.

Console

Because EMR Notebooks are EMR Studio Workspaces in the new console, you can you can refer to Link Git-based repositories to an EMR Studio Workspace for more information on working with Git repositories in your Workspace. But at this time, you can't delete Git repositories from Workspaces.

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 in EMR Studio.

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 in EMR Studio

> **Note**
>
> EMR Notebooks are available as EMR Studio Workspaces in the console. The **Create Workspace** button in the console lets you create new notebooks. To access or create Workspaces, EMR Notebooks users need additional IAM role permissions. For more information, see Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console and Amazon EMR console.

To create a notebook and associate it with Git repositories in the old Amazon EMR console

1. Follow the instructions at Create a Notebook in EMR Studio.

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. We recommend 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 an EMR Studio Notebook

> **Note**
>
> EMR Notebooks are available as EMR Studio Workspaces in the console. The **Create Workspace** button in the console lets you create new notebooks. To access or create Workspaces, EMR Notebooks users need additional IAM role permissions. For more information, see Amazon EMR Notebooks are Amazon EMR Studio Workspaces in the console and Amazon EMR console.

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 primary
```

To open any of the additional repositories, navigate to other folders.

If you choose to open the notebook with a JupyterLab interface, you can use the pre-installed JupyterLab Git extension. For information about the extension, see jupyterlab-git.

Plan, configure and launch Amazon EMR 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 Amazon EMR cluster location and data storage.
- Whether you are running Amazon EMR clusters on Outposts or Local Zones. See EMR clusters on AWS Outposts or EMR clusters on AWS Local Zones.
- Whether a cluster is long-running or transient, and what software it runs. See Configuring an Amazon EMR cluster to continue or terminate after step execution and Configure applications when you launch your Amazon EMR cluster.
- Whether a cluster has a single primary node or three primary nodes. See Plan and configure primary nodes in your Amazon EMR cluster.
- The hardware and networking options that optimize cost, performance, and availability for your application. See Configure Amazon EMR cluster hardware and networking.
- How to set up clusters so you can manage them more easily, and monitor activity, performance, and health. See Configure Amazon EMR cluster logging and debugging and Tag and categorize Amazon EMR cluster resources.
- How to authenticate and authorize access to cluster resources, and how to encrypt data. See Security in Amazon EMR.
- How to integrate with other software and services. See Drivers and third-party application integration on Amazon EMR.

Launch an Amazon EMR cluster quickly

Follow these steps to launch an Amazon EMR cluster in just a few minutes.

To quickly launch a cluster with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr/clusters.

2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose **Create cluster**.
3. On the **Create Cluster** page, enter or select values for the provided fields. The persistent summary panel displays a real-time view of your currently selected cluster options. Select a heading in the summary panel to navigate to the corresponding section and make adjustments. Your cluster name can't contain the characters <, >, $, |, or ` (backtick). You must complete all required configurations before you can choose **Create cluster**.
4. Choose **Create cluster** to accept the configuration as shown.
5. The cluster details page opens. 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 might need to choose the refresh icon on the upper 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 Amazon EMR 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 for your Amazon EMR cluster
- Working with storage and file systems with Amazon EMR
- Prepare input data for processing with Amazon EMR
- Configure a location for Amazon EMR cluster output

Choose an AWS Region for your Amazon EMR cluster

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 with the console

Your default Region is displayed to the left of your account information on the navigation bar. To switch Regions in both the new and old consoles, choose the Region dropdown menu and select a new option.

Specify a Region with the AWS CLI

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 with 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.

Working with storage and file systems with Amazon EMR

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://amzn-s3-demo-bucket1/path` references an Amazon S3 bucket using S3A (since EMR-7.10.0 release). 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 S3A 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.
S3A	`s3://`, `s3a://`, `s3n://`	The Hadoop S3A filesystem is a open source S3 connector that enables Apache Hadoop and its ecosystem to interact directly with Amazon S3 storage. It allows users to read and write data to S3 buckets using Hadoop-compatible file operations, providing a seamless integration between Hadoop applications and cloud storage. **Note** Prior to EMR-7.10.0, Amazon EMR used EMRFS for *s3://* and *s3n://* scheme.
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

File system	Prefix	Description
		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. With the multipart upload functionality Amazon EMR provides through the AWS Java SDK, you can upload large files to the Amazon S3 native file system, and the Amazon S3 block file system is deprecated. For more information about multipart upload for EMR, see Configure multipart upload for Amazon S3. For more information about S3 object-size and part-size limits, see Amazon S3 multipart upload limits in the **Amazon Simple Storage Service** *User Guide*.

  > **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 for processing with Amazon EMR

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
- Different ways to get data into Amazon EMR

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.

Different ways 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
- Upload data with AWS DataSync
- Import files with distributed cache with Amazon EMR
- Detecting and processing compressed files with Amazon EMR
- Import DynamoDB data into Hive with Amazon EMR
- Connect to data with AWS Direct Connect from Amazon EMR
- Upload large amounts of data for Amazon EMR with AWS Snowball Edge

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
- Configure multipart upload for Amazon S3
- Best practices
- Upload data to Amazon S3 Express One Zone

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.

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 uploads

Console

To disable multipart uploads with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose **Create cluster**.
3. Under **Software settings**, enter the following configuration: `classification=core-site,properties=[fs.s3n.multipart.uploads.enabled=false]`.

4. Choose any other options that apply to your cluster.
5. To launch your cluster, choose **Create cluster**.

CLI

To 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.

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-7.10.0 --applications Name=Hive Name=Pig \
--use-default-roles --ec2-attributes KeyName=myKey --instance-type m5.xlarge \
--instance-count 3 --configurations file://myConfig.json
```

API

To 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. 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.

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 to Amazon S3 Express One Zone

Overview

With Amazon EMR 6.15.0 and higher, you can use Amazon EMR with Apache Spark in conjunction with the Amazon S3 Express One Zone storage class for improved performance on your Spark jobs. Amazon EMR releases 7.2.0 and higher also support HBase, Flink, and Hive, so you can also benefit from S3 Express One Zone if you use these applications. *S3 Express One Zone* is an S3 storage class for applications that frequently access data with hundreds of thousands of requests per second. At the time of its release, S3 Express One Zone delivers the lowest latency and highest performance cloud object storage in Amazon S3.

Prerequisites

- **S3 Express One Zone permissions** – When S3 Express One Zone initially performs an action like `GET`, `LIST`, or `PUT` on an S3 object, the storage class calls `CreateSession` on your behalf. Your IAM policy must allow the `s3express:CreateSession` permission so that the S3A connector can invoke the `CreateSession` API. For an example policy with this permission, see Getting started with Amazon S3 Express One Zone.

- **S3A connector** – To configure your Spark cluster to access data from an Amazon S3 bucket that uses the S3 Express One Zone storage class, you must use the Apache Hadoop connector S3A. To use the connector, ensure all S3 URIs use the `s3a` scheme. If they don't, you can change the filesystem implementation that you use for `s3` and `s3n` schemes.

To change the `s3` scheme, specify the following cluster configurations:

```
[
  {
    "Classification": "core-site",
    "Properties": {
      "fs.s3.impl": "org.apache.hadoop.fs.s3a.S3AFileSystem",
      "fs.AbstractFileSystem.s3.impl": "org.apache.hadoop.fs.s3a.S3A"
    }
  }
]
```

To change the `s3n` scheme, specify the following cluster configurations:

```
[
  {
    "Classification": "core-site",
    "Properties": {
      "fs.s3n.impl": "org.apache.hadoop.fs.s3a.S3AFileSystem",
      "fs.AbstractFileSystem.s3n.impl": "org.apache.hadoop.fs.s3a.S3A"
    }
  }
]
```

Getting started with Amazon S3 Express One Zone

Topics

- Create a permission policy
- Create and configure your cluster
- Configurations overview

Create a permission policy

Before you can create a cluster that uses Amazon S3 Express One Zone, you must create an IAM policy to attach to the Amazon EC2 instance profile for the cluster. The policy must have permissions to access the S3 Express One Zone storage class. The following example policy shows how to grant the required permission. After you create the policy, attach the policy to the instance profile role that you use to create your EMR cluster, as described in the Create and configure your cluster section.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3express:*:123456789012:bucket/example-s3-bucket"
      ],
      "Action": [
        "s3express:CreateSession"
      ],
      "Sid": "AllowS3EXPRESSCreatesession"
    }
  ]
}
```

Create and configure your cluster

Next, create a cluster that runs Spark, HBase, Flink, or Hive with S3 Express One Zone. The following steps describe a high-level overview to create a cluster in the AWS Management Console:

1. Navigate to the Amazon EMR console and select **Clusters** from the sidebar. Then choose **Create cluster**.
2. If you use Spark, select Amazon EMR release `emr-6.15.0` or higher. If you use HBase, Flink, or Hive, select `emr-7.2.0` or higher.
3. Select the applications that you want to include on your cluster, such as Spark, HBase, or Flink.

4. To enable Amazon S3 Express One Zone, enter a configuration similar to the following example in the **Software settings** section. The configurations and recommended values are described in the Configurations overview section that follows this procedure.

```
[
  {
    "Classification": "core-site",
    "Properties": {
      "fs.s3a.aws.credentials.provider":
 "software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider",
      "fs.s3a.change.detection.mode": "none",
      "fs.s3a.endpoint.region": "aa-example-1",
      "fs.s3a.select.enabled": "false"
    }
  },
  {
    "Classification": "spark-defaults",
    "Properties": {
      "spark.sql.sources.fastS3PartitionDiscovery.enabled": "false"
    }
  }
]
```

5. In the **EC2 instance profile for Amazon EMR** section, choose to use an existing role, and use a role with the policy attached that you created in the Create a permission policy section above.
6. Configure the rest of your cluster settings as appropriate for your application, and then select **Create cluster**.

Configurations overview

The following tables describe the configurations and suggested values that you should specify when you set up a cluster that uses S3 Express One Zone with Amazon EMR, as described in the Create and configure your cluster section.

S3A configurations

Parameter	Default value	Suggested value	Explanation
	If not specified, uses `AWSCreden`	`software.amazon.aw`	The Amazon EMR instance profile role

Parameter	Default value	Suggested value	Explanation
`fs.s3a.aws.credentials.provider`	`tialProviderList` in the following order: `Temporary AWSCredentialsProvider`, `SimpleAWSCredentialsProvider`, `EnvironmentVariableCredentialsProvider`, `IAMInstanceCredentialsProvider`.	`ssdk.auth.credentials.InstanceProfileCredentialsProvider`	should have the policy that allows the S3A filesystem to call `s3express:CreateSession`. Other crendential providers also work if they have the S3 Express One Zone permissions.
`fs.s3a.endpoint.region`	null	The AWS Region where you created the bucket.	Region resolution logic doesn't work with S3 Express One Zone storage class.
`fs.s3a.select.enabled`	`true`	`false`	Amazon S3 `select` is not supported with S3 Express One Zone storage class.

Parameter	Default value	Suggested value	Explanation
`fs.s3a.change.detection.mode`	`server`	none	Change detection by S3A works by checking MD5-based `etags`. S3 Express One Zone storage class doesn't support MD5 `checksums`.

Spark configurations

Parameter	Default value	Suggested value	Explanation
`spark.sql.sources.fastS3PartitionDiscovery.enabled`	`true`	`false`	The internal optimization uses an S3 API parameter that the S3 Express One Zone storage class doesn't support.

Hive configurations

Parameter	Default value	Suggested value	Explanation
`hive.exec.fast.s3.partition.discovery.enabled`	`true`	`false`	The internal optimization uses an S3 API parameter that the S3 Express One Zone storage class doesn't support.

Considerations

Consider the following when you integrate Apache Spark on Amazon EMR with the S3 Express One Zone storage class:

- The S3A connector is required to use S3 Express One Zone with Amazon EMR. Only S3A has the features and storage classes that are required to interact with S3 Express One Zone. For steps to set up the connector, see the section called "Prerequisites".
- The Amazon S3 Express One Zone storage class supports SSE-S3 and SSE-KMS encryption. For more information, see Server-side encryption with Amazon S3.
- The Amazon S3 Express One Zone storage class does not support writes with the S3A `FileOutputCommitter`. Writes with the S3A `FileOutputCommitter` on S3 Express One Zone buckets result in an error: *InvalidStorageClass: The storage class you specified is not valid*.
- Amazon S3 Express One Zone is supported with Amazon EMR releases 6.15.0 and higher on EMR on EC2. Additionally, it's supported on Amazon EMR releases 7.2.0 and higher on Amazon EMR on EKS and on Amazon EMR Serverless.

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 with Amazon EMR

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 with an Amazon EMR cluster.

Topics

- Supported file types
- Location of cached files
- Access cached files from streaming applications
- Access cached files from streaming applications

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 primary 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

You can use the AWS Management Console and the AWS CLI to create clusters that use Distributed Cache.

Console

To specify distributed cache files with the new console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose **Create cluster**.
3. Under **Steps**, choose **Add step**. This opens the **Add step** dialog. In the **Arguments** field, include the files and archives to save to the cache. 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 add an individual file to the distributed cache, specify `-cacheFile`, followed by the name and location of the file, the pound (#) sign, and the name you want to give the file when it's placed in the local cache. The following example demonstrates how to add an individual file to the distributed cache.

   ```
   -cacheFile \
   s3://amzn-s3-demo-bucket/file-name#cache-file-name
   ```

   If you want to 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. The following example demonstrates how to add an archive file to the distributed cache.

   ```
   -cacheArchive \
   s3://amzn-s3-demo-bucket/archive-name#cache-archive-name
   ```

   Enter appropriate values in the other dialog fields. Options differ depending on the step type. To add your step and exit the dialog, choose **Add step**.
4. Choose any other options that apply to your cluster.
5. To launch your cluster, choose **Create cluster**.

CLI

To specify distributed cache files with 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 an individual file to the distributed cache, specify `-cacheFile`, followed by the name and location of the file, the pound (#) sign, and the name you want to give the file when it's placed in the local cache.

  If you want to 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. The following example demonstrates how to add an archive file to the distributed 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 primary 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 primary 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.

Detecting and processing compressed files with Amazon EMR

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 with Amazon EMR

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 from Amazon EMR

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 for Amazon EMR with AWS Snowball Edge

AWS Snowball Edge 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 Edge 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 Edge devices secure and protect your data while regional shipping carriers transport them between Amazon S3 and your onsite data storage location. Snowball Edge devices are physically rugged and protected by the AWS Key Management Service (AWS KMS). For more information, see the AWS Snowball Edge Edge Developer Guide.

Configure a location for Amazon EMR cluster output

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
- What formats can Amazon EMR return?
- How to write data to an Amazon S3 bucket you don't own with Amazon EMR
- Ways to compress the output of your Amazon EMR cluster

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://amzn-s3-demo-bucket1/script/MapperScript.py`
log files	`s3://amzn-s3-demo-bucket1/logs`
input data	`s3://amzn-s3-demo-bucket1/input`

Information	Example Location on Amazon S3
output data	`s3://amzn-s3-demo-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 with Amazon EMR

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.
`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 Amazon EMR cluster primary node using SSH.

  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://amzn-s3-demo-bucket/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 Amazon EMR cluster primary node using SSH.

  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://amzn-s3-demo-bucket/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://amzn-s3-demo-bucket/input s3://amzn-
s3-demo-bucket/output
```

Ways to compress the output of your Amazon EMR cluster

There are different ways to compress output that results from data processing. The compression tools you use depend on properties of your data. Compression can improve performance when you transfer large amounts of data.

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 primary nodes in your Amazon EMR cluster

When you launch an Amazon EMR cluster, you can choose to have one or three primary nodes in your cluster. High availability for *instance fleets* is supported with Amazon EMR releases 5.36.1, 5.36.2, 6.8.1, 6.9.1, 6.10.1, 6.11.1, 6.12.0, and higher. For *instance groups*, high availability is supported with Amazon EMR releases 5.23.0 and higher. To further improve cluster availability, Amazon EMR can use Amazon EC2 placement groups to ensure that primary nodes are placed on distinct underlying hardware. For more information, see Amazon EMR integration with EC2 placement groups.

An Amazon EMR cluster with multiple primary nodes provides the following benefits:

- The primary node is no longer a single point of failure. If one of the primary nodes fails, the cluster uses the other two primary nodes and runs without interruption. In the meantime, Amazon EMR automatically replaces the failed primary 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 Amazon EMR cluster with multiple primary nodes supports open source applications and other Amazon EMR features, see Features that support high availability in an Amazon EMR cluster and how they work with open-source applications.

> **Note**
>
> The cluster can reside only in one Availability Zone or subnet.

This section provides information about supported applications and features of an Amazon EMR cluster with multiple primary nodes as well as the configuration details, best practices, and considerations for launching the cluster.

Topics

- Features that support high availability in an Amazon EMR cluster and how they work with open-source applications
- Launch an Amazon EMR Cluster with multiple primary nodes
- Amazon EMR integration with EC2 placement groups
- Considerations and best practices when you create an Amazon EMR cluster with multiple primary nodes

Features that support high availability in an Amazon EMR cluster and how they work with open-source applications

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 Amazon EMR cluster with multiple primary 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 upto 5.36.2, only two of the three primary nodes run HDFS NameNode.
> In Amazon EMR versions 6.x and higher, all three of the primary nodes run HDFS NameNode.

If you need to find out which NameNode is `active`, you can use SSH to connect to any primary 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 Amazon EMR cluster with multiple primary nodes enables the YARN ResourceManager high availability feature in Hadoop. For more information, see ResourceManager high availability.

In an Amazon EMR cluster with multiple primary nodes, YARN ResourceManager runs on all three primary nodes. One ResourceManager is in `active` state, and the other two are in `standby` state. If the primary node with `active` ResourceManager fails, Amazon EMR starts an automatic failover

process. A primary node with a `standby` ResourceManager takes over all operations. Amazon EMR replaces the failed primary 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 primary node, which automatically directs you to the `active` resource manager. To find out which resource manager is `active`, use SSH to connect to any primary node in the cluster. Then run the following command to get a list of the three primary nodes and their status:

```
yarn rmadmin -getAllServiceState
```

Supported applications in an Amazon EMR Cluster with multiple primary nodes

You can install and run the following applications on an Amazon EMR cluster with multiple primary nodes. For each application, the primary node failover process varies.

Application	Availability during primary node failover	Notes
Flink	Availability not affected by primary 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 primary 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 primary nodes.

Application	Availability during primary node failover	Notes
		Beginning with Amazon EMR version 5.28.0, no manual configuration is needed to enable JobManager high availability.
Ganglia	Availability not affected by primary node failover	Ganglia is available on all primary nodes, so Ganglia can continue to run during the primary node failover process.
Hadoop	High availability	HDFS NameNode and YARN ResourceManager automatically fail over to the standby node when the active primary node fails.
HBase	High availability	HBase automatically fails over to the standby node when the active primary node fails. If you are connecting to HBase through a REST or Thrift server, you must switch to a different primary node when the active primary node fails.
HCatalog	Availability not affected by primary node failover	HCatalog is built upon Hive metastore, which exists outside of the cluster. HCatalog remains available during the primary node failover process.
JupyterHub	High availability	JupyterHub is installed on all three primary instances. It is highly recommended to configure notebook persistence to preven t notebook loss upon primary node failure. For more information, see Configuring persi stence for notebooks in Amazon S3.

Application	Availability during primary node failover	Notes
Livy	High availability	Livy is installed on all three primary nodes. When the active primary node fails, you lose access to the current Livy session and need to create a new Livy session on a different primary node or on the new replacement node.
Mahout	Availability not affected by primary node failover	Since Mahout has no daemon, it is not affected by the primary node failover process.
MXNet	Availability not affected by primary node failover	Since MXNet has no daemon, it is not affected by the primary node failover process.
Phoenix	High Availability	Phoenix' QueryServer runs only on one of the three primary 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 primary node failover	Since Pig has no daemon, it is not affected by the primary node failover process.
Spark	High availability	All Spark applications run in YARN containers and can react to primary 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

Application	Availability during primary node failover	Notes
Tez	High availability	Since Tez containers run on YARN, Tez behaves the same way as YARN during the primary node failover process.
TensorFlow	Availability not affected by primary node failover	Since TensorFlow has no daemon, it is not affected by the primary node failover process.
Zeppelin	High availability	Zeppelin is installed on all three primary nodes. Zeppelin stores notes and interpreter configurations in HDFS by default to prevent data loss. Interpreter sessions are completely isolated across all three primary instances. Session data will be lost upon master failure. It is recommended to not modify the same note concurrently on different primary instances.
ZooKeeper	High availability	ZooKeeper is the foundation of the HDFS automatic failover feature. ZooKeeper provides a highly available service for maint aining 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 primary nodes, you must configure an external database. The external database exists outside the cluster and makes data persistent during the primary node failover process. For the following applications, the service components will automatically recover during the primary node failover process, but active jobs may fail and need to be retried.

Application	Availability during primary 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 primary 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 primary nodes so you can use it to access the Presto Coordinator from any of the primary nodes. The Presto Coordinator is installed on only one primary node. You can find the DNS name of the primary node where the Presto Coordinator is installed by calling the Amazon

Application	Availability during primary node failover	Notes
		EMR `describe-cluster` API and reading the returned value of the `MasterPublicDnsName` field in the response.

> **Note**
>
> When a primary node fails, your Java Database Connectivity (JDBC) or Open Database Connectivity (ODBC) terminates its connection to the primary node. You can connect to any of the remaining primary nodes to continue your work because the Hive metastore daemon runs on all primary nodes. Or you can wait for the failed primary node to be replaced.

How Amazon EMR features work in a cluster with multiple primary nodes

Connecting to primary nodes using SSH

You can connect to any of the three primary nodes in an Amazon EMR cluster using SSH in the same way you connect to a single primary node. For more information, see Connect to the primary node using SSH.

If a primary node fails, your SSH connection to that primary node ends. To continue your work, you can connect to one of the other two primary nodes. Alternatively, you can access the new primary node after Amazon EMR replaces the failed one with a new one.

> **Note**
>
> The private IP address for the replacement primary node remains the same as the previous one. The public IP address for the replacement primary 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 primary nodes. However, you can run `hdfs` CLI commands and operate jobs to access HDFS on all three primary nodes.

Working with steps in an Amazon EMR Cluster with multiple primary nodes

You can submit steps to an Amazon EMR cluster with multiple primary nodes in the same way you work with steps in a cluster with a single primary node. For more information, see Submit work to a cluster.

The following are considerations for working with steps in an Amazon EMR cluster with multiple primary nodes:

- If a primary node fails, the steps that are running on the primary 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 primary node fails, the step can continue and succeed due to the automatic failover of the primary 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 primary 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. To shut down a cluster with multiple primary nodes, you must first modify the cluster attributes to disable termination protection. For instructions, see Terminate an Amazon EMR Cluster with multiple primary nodes.

For more information about termination protection, see Using termination protection to protect your Amazon EMR clusters from accidental shut down.

Unsupported features in an Amazon EMR Cluster with multiple primary nodes

The following Amazon EMR features are currently not available in an Amazon EMR cluster with multiple primary nodes:

- EMR Notebooks
- 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 primary nodes or for Amazon 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 Amazon EMR cluster with multiple primary nodes. For more information, see Transparent encryption in HDFS on Amazon EMR.

Launch an Amazon EMR Cluster with multiple primary nodes

This topic provides configuration details and examples for launching an Amazon EMR cluster with multiple primary nodes.

> **Note**
>
> Amazon EMR automatically enables termination protection for all clusters that have multiple primary nodes, and overrides any auto-termination settings that you supply when you create the cluster. To shut down a cluster with multiple primary nodes, you must first modify the cluster attributes to disable termination protection. For instructions, see Terminate an Amazon EMR Cluster with multiple primary nodes.

Prerequisites

- You can launch an Amazon EMR cluster with multiple primary nodes in both public and private VPC subnets. **EC2-Classic** is not supported. To launch an Amazon EMR cluster with multiple primary 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 Amazon EMR cluster with multiple primary 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 primary nodes

You can launch a cluster with multiple primary nodes when you use instance groups or instance fleets. When you use *instance groups* with multiple primary nodes, you must specify an instance count value of `3` for the primary node instance group. When you use *instance fleets* with multiple primary nodes, you must specify the `TargetOnDemandCapacity` of `3`, `TargetSpotCapacity` of `0` for the primary instance fleet, and `WeightedCapacity` of `1` for each instance type that you configure for the primary fleet.

The following examples demonstrate how to launch the cluster using the default AMI or a custom AMI with both instance groups and instance fleets:

> **Note**
>
> You must specify the subnet ID when you launch an Amazon EMR cluster with multiple primary nodes using the AWS CLI. Replace *`22XXXX01`* and *`22XXXX02`* with your subnet ID in the following examples.

Default AMI, instance groups

Example Example – Launching an Amazon EMR instance group cluster with multiple primary nodes using a default AMI

```
aws emr create-cluster \
--name "ha-cluster" \
--release-label emr-6.15.0 \
--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
```

Default AMI, instance fleets

Example Example – Launching an Amazon EMR instance fleet cluster with multiple primary nodes using a default AMI

```
aws emr create-cluster \
--name "ha-cluster" \
--release-label emr-6.15.0 \
--instance-fleets '[
    {
        "InstanceFleetType": "MASTER",
        "TargetOnDemandCapacity": 3,
        "TargetSpotCapacity": 0,
        "LaunchSpecifications": {
            "OnDemandSpecification": {
                "AllocationStrategy": "lowest-price"
            }
        },
        "InstanceTypeConfigs": [
            {
                "WeightedCapacity": 1,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.xlarge"
            },
            {
                "WeightedCapacity": 1,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.2xlarge"
            },
            {
                "WeightedCapacity": 1,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.4xlarge"
            }
        ],
        "Name": "Master - 1"
    },
    {
        "InstanceFleetType": "CORE",
        "TargetOnDemandCapacity": 5,
        "TargetSpotCapacity": 0,
```

```
        "LaunchSpecifications": {
            "OnDemandSpecification": {
                "AllocationStrategy": "lowest-price"
            }
        },
        "InstanceTypeConfigs": [
            {
                "WeightedCapacity": 1,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.xlarge"
            },
            {
                "WeightedCapacity": 2,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.2xlarge"
            },
            {
                "WeightedCapacity": 4,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.4xlarge"
            }
        ],
        "Name": "Core - 2"
    }
]' \
 --ec2-attributes '{"InstanceProfile":"EMR_EC2_DefaultRole","SubnetIds":
["subnet-22XXXX01", "subnet-22XXXX02"]}' \
--service-role EMR_DefaultRole \
--applications Name=Hadoop Name=Spark
```

Custom AMI, instance groups

Example Example – Launching an Amazon EMR instance group cluster with multiple primary nodes using a custom AMI

```
aws emr create-cluster \
--name "custom-ami-ha-cluster" \
--release-label emr-6.15.0 \
--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
```

Custom AMI, instance fleets

Example Example – Launching an Amazon EMR instance fleet cluster with multiple primary nodes using a custom AMI

```
aws emr create-cluster \
--name "ha-cluster" \
--release-label emr-6.15.0 \
--instance-fleets '[
    {
        "InstanceFleetType": "MASTER",
        "TargetOnDemandCapacity": 3,
        "TargetSpotCapacity": 0,
        "LaunchSpecifications": {
            "OnDemandSpecification": {
                "AllocationStrategy": "lowest-price"
            }
        },
        "InstanceTypeConfigs": [
            {
                "WeightedCapacity": 1,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.xlarge"
            },
            {
                "WeightedCapacity": 1,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.2xlarge"
            },
            {
                "WeightedCapacity": 1,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.4xlarge"
            }
        ],
        "Name": "Master - 1"
    },
    {
        "InstanceFleetType": "CORE",
        "TargetOnDemandCapacity": 5,
```

```
        "TargetSpotCapacity": 0,
        "LaunchSpecifications": {
            "OnDemandSpecification": {
                "AllocationStrategy": "lowest-price"
            }
        },
        "InstanceTypeConfigs": [
            {
                "WeightedCapacity": 1,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.xlarge"
            },
            {
                "WeightedCapacity": 2,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.2xlarge"
            },
            {
                "WeightedCapacity": 4,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.4xlarge"
            }
        ],
        "Name": "Core - 2"
    }
]' \
--ec2-attributes '{"InstanceProfile":"EMR_EC2_DefaultRole","SubnetIds":
["subnet-22XXXX01", "subnet-22XXXX02"]}' \
--service-role EMR_DefaultRole \
--applications Name=Hadoop Name=Spark \
--custom-ami-id ami-MyAmiID
```

Terminate an Amazon EMR Cluster with multiple primary nodes

To terminate an Amazon EMR cluster with multiple primary 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 primary node cluster on Amazon EC2, you have the option to use placement group strategies to specify how you want the primary 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 primary node clusters. Currently, only primary node types are supported by the placement group strategy, and the `SPREAD` strategy is applied to those primary nodes. The `SPREAD` strategy places a small group of instances across separate underlying hardware to guard against the loss of multiple primary 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.

Attach the placement group managed policy to the Amazon EMRrole

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 cluster with multiple primary nodes.

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).

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.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Resource": [
        "*"
      ],
      "Effect": "Allow",
      "Action": [
        "ec2:DeletePlacementGroup",
        "ec2:DescribePlacementGroups"
      ],
      "Sid": "AllowEC2Deleteplacementgroup"
    },
    {
      "Resource": [
        "arn:aws:ec2:*:*:placement-group/pg-*"
      ],
      "Effect": "Allow",
      "Action": [
        "ec2:CreatePlacementGroup"
      ],
      "Sid": "AllowEC2Createplacementgroup"
    }
  ]
}
```

Launch an Amazon EMR cluster with multiple primary nodes using placement group strategy

To launch an Amazon EMR cluster that has multiple primary nodes with a placement group strategy, attach the placement group managed policy `AmazonElasticMapReducePlacementGroupPolicy` to the Amazon EMR role. For more information, see Attach the placement group managed policy to the Amazon EMRrole.

Every time you use this role to start an Amazon EMR cluster with multiple primary nodes, Amazon EMR attempts to launch a cluster with `SPREAD` strategy applied to its primary 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 cluster that has multiple primary nodes without a placement group strategy.

If you launch an Amazon EMR cluster that has multiple primary nodes with the `placement-group-configs` parameter using the Amazon EMRAPI or CLI, Amazon EMR only launches the cluster if the Amazon EMRrole has the placement group managed policy `AmazonElasticMapReducePlacementGroupPolicy` attached. If the Amazon EMRrole does not have the policy attached, the Amazon EMR cluster with multiple primary nodes start fails.

Amazon EMR API

Example Example – Use a placement group strategy to launch an instance group cluster with multiple primary nodes from the Amazon EMR API

When you use the RunJobFlow action to create an Amazon EMR cluster with multiple primary nodes, 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-6.15.0,
   "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.

AWS CLI

Example Example – Use a placement group strategy to launch an instance fleet cluster with multiple primary nodes from the AWS Command Line Interface

When you use the RunJobFlow action to create an Amazon EMR cluster with multiple primary nodes, set the `PlacementGroupConfigs` property to the following. Currently, the `MASTER` instance role automatically uses `SPREAD` as the placement group strategy.

```
aws emr create-cluster \
--name "ha-cluster" \
--placement-group-configs InstanceRole=MASTER \
--release-label emr-6.15.0 \
--instance-fleets '[
    {
        "InstanceFleetType": "MASTER",
        "TargetOnDemandCapacity": 3,
        "TargetSpotCapacity": 0,
        "LaunchSpecifications": {
```

```
            "OnDemandSpecification": {
                "AllocationStrategy": "lowest-price"
            }
        },
        "InstanceTypeConfigs": [
            {
                "WeightedCapacity": 1,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.xlarge"
            },
            {
                "WeightedCapacity": 1,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.2xlarge"
            },
            {
                "WeightedCapacity": 1,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.4xlarge"
            }
        ],
        "Name": "Master - 1"
    },
    {
        "InstanceFleetType": "CORE",
        "TargetOnDemandCapacity": 5,
        "TargetSpotCapacity": 0,
        "LaunchSpecifications": {
            "OnDemandSpecification": {
                "AllocationStrategy": "lowest-price"
            }
        },
        "InstanceTypeConfigs": [
            {
                "WeightedCapacity": 1,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.xlarge"
            },
            {
                "WeightedCapacity": 2,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.2xlarge"
            },
            {
```

```
                "WeightedCapacity": 4,
                "BidPriceAsPercentageOfOnDemandPrice": 100,
                "InstanceType": "m5.4xlarge"
            }
        ],
        "Name": "Core - 2"
    }
]' \
--ec2-attributes '{
    "KeyName": "ec2_key_pair_name",
    "InstanceProfile": "EMR_EC2_DefaultRole",
    "SubnetIds": [
        "subnet-22XXXX01",
        "subnet-22XXXX02"
    ]
}' \
--service-role EMR_DefaultRole \
--applications Name=Hadoop Name=Spark
```

- Replace *ha-cluster* with the name of your high-availability cluster.
- 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.
- Replace *subnet-22XXXX01* and *subnet-22XXXX02*with your subnet IDs.

Launch a cluster with multiple primary nodes without a placement group strategy

For a cluster with multiple primary nodes to launch primary nodes without the placement group strategy, you need to do one of the following:

- Remove the placement group managed policy `AmazonElasticMapReducePlacementGroupPolicy`from the Amazon EMRrole, or
- Launch a cluster with multiple primary nodes with the `placement-group-configs` parameter using the Amazon EMRAPI or CLI choosing `NONE` as the placement group strategy.

Amazon EMR API

Example – Launching a cluster with multiple primary nodes without placement group strategy using the Amazon EMRAPI.

When using the RunJobFlow action to create a cluster with multiple primary nodes, 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.

Amazon EMR CLI

Example – Launching a cluster with multiple primary nodes without a placement group strategy using the Amazon EMRCLI.

When using the RunJobFlow action to create a cluster with multiple primary nodes, set the `PlacementGroupConfigs` property to the following.

```
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 cluster with multiple primary nodes

You can use the Amazon EMR describe cluster API to see the placement group strategy configuration attached to the cluster with multiple primary nodes.

Example

```
aws emr describe-cluster --cluster-id "j-xxxxx"
{
   "Cluster":{
      "Id":"j-xxxxx",
      ...
      ...
      "PlacementGroups":[
```

```
            {
               "InstanceRole":"MASTER",
               "PlacementStrategy":"SPREAD"
            }
         ]
      }
   }
```

Considerations and best practices when you create an Amazon EMR cluster with multiple primary nodes

Consider the following when you create an Amazon EMR cluster with multiple primary nodes:

> **Important**
>
> To launch high-availability EMR clusters with multiple primary nodes, we strongly recommend that you use the latest Amazon EMR release. This ensures that you get the highest level of resiliency and stability for your high-availability clusters.

- High availability for *instance fleets* is supported with Amazon EMR releases 5.36.1, 5.36.2, 6.8.1, 6.9.1, 6.10.1, 6.11.1, 6.12.0, and higher. For *instance groups*, high availability is supported with Amazon EMR releases 5.23.0 and higher. To learn more, see About Amazon EMR Releases.
- On high-availability clusters, Amazon EMR only supports the launch of primary nodes with On Demand instances. This ensures the highest availability for your cluster.
- You can still specify multiple instance types for primary fleet but all the primary nodes of high-availability clusters are launched with the same instance type, including replacements for unhealthy primary nodes.
- To continue operations, a high-availability cluster with multiple primary nodes requires two out of three primary nodes to be healthy. As a result, if any two primary nodes fail simultaneously, your EMR cluster will fail.
- All EMR clusters, including high-availability clusters, are launched in a single Availability Zone. Therefore, they can't tolerate Availability Zone failures. In the case of an Availability Zone outage, you lose access to the cluster.
- If you use If you're using a custom service role or policy when you launch a cluster inside an instance fleet, you can add the `ec2:DescribeInstanceTypeOfferings` permission so

Amazon EMR can filter out unsupported Availability Zones (AZ). When Amazon EMR filters out the AZs that don't support any instance types of primary nodes, Amazon EMR prevents cluster launches from failing because of unsupported primary instance types. For more information, see Instance type not supported.

- Amazon EMR doesn't guarantee high availability for open-source applications other than the ones that are specified in Supported applications in an Amazon EMR Cluster with multiple primary nodes.
- In Amazon EMR releases 5.23.0 through 5.36.2, only two of the three primary nodes for an instance group cluster run HDFS NameNode.
- In Amazon EMR releases 6.x and higher, all three of the primary nodes for an instance group run HDFS NameNode.

Considerations for configuring subnet:

- An Amazon EMR cluster with multiple primary nodes can reside only in one Availability Zone or subnet. Amazon EMR cannot replace a failed primary 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 Amazon 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 nodes are also highly available, we recommend that you launch at least four core nodes. If you decide to launch a smaller cluster with three or fewer core nodes, set `dfs.replication parameter` to at least 2 for HDFS to have sufficient DFS replication. For more information, see HDFS configuration.

Warning

1. Setting `dfs.replication` to 1 on clusters with fewer than four nodes can lead to HDFS data loss if a single node goes down. We recommend you use a cluster with at least four core nodes for production workloads.
2. Amazon EMR will not allow clusters to scale core nodes below `dfs.replication`. For example, if `dfs.replication = 2`, the minimum number of core nodes is 2.

3. When you use Managed Scaling, Auto-scaling, or choose to manually resize your cluster, we recommend that you to set `dfs.replication` to 2 or higher.

Considerations for Setting Alarms on Metrics:

- Amazon EMR doesn't provide application-specific metrics about HDFS or YARN. We reccommend that you set up alarms to monitor the primary node instance count. Configure the alarms using the following Amazon CloudWatch metrics: `MultiMasterInstanceGroupNodesRunning`, `MultiMasterInstanceGroupNodesRunningPercentage`, or `MultiMasterInstanceGroupNodesRequested`. CloudWatch will notify you in the case of primary node failure and replacement.
  - If the `MultiMasterInstanceGroupNodesRunningPercentage` is lower than 100% and greater than 50%, the cluster may have lost a primary node. In this situation, Amazon EMR attempts to replace a primary node.
  - If the `MultiMasterInstanceGroupNodesRunningPercentage` drops below 50%, two primary nodes may have failed. In this situation, the quorum is lost and the cluster can't be recovered. You must manually migrate data off of this cluster.

  For more information, see Setting alarms on metrics.

EMR clusters on AWS Outposts

Beginning with Amazon EMR 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 Amazon 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.
- When you use AWS Outposts with Amazon EMR releases 5.28 through 6.x, you can only use S3 buckets that store objects in an AWS Region that you specify. With Amazon EMR 7.0.0 and higher, Amazon EMR on AWS Outposts is also supported with the S3A filesystem client, prefix `s3a://`.
- 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` | `c5.18xlarge` | `c5d.xlarge` | `c5d.2xlarge` | `c5d.4xlarge` | `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 Amazon EMR cluster with multiple primary 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.

Console

To create a cluster on AWS Outposts with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose **Create cluster**.
3. Under **Cluster configuration**, select **Instance groups** or **Instance fleets**. Then, choose an instance type from the **Choose EC2 instance type** dropdown menu or select **Actions** and choose **Add EBS volumes**. Amazon EMR on AWS Outposts supports limited Amazon EBS volume and instance types.
4. Under **Networking**, select an EC2 subnet with an Outpost ID in this format: op-123456789.
5. Choose any other options that apply to your cluster.
6. To launch your cluster, choose **Create cluster**.

CLI

To create a cluster on AWS Outposts with the 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, as in the following example. Replace *subnet-22XXXX01* with your own Amazon EC2 subnet ID.

```
aws emr create-cluster \
--name "Outpost cluster" \
--release-label emr-7.10.0 \
--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).

Console

To create a cluster on a Local Zone with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.

2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose **Create cluster**.
3. Under **Networking**, select an EC2 subnet with a Local Zone ID in this format: subnet 123abc | us-west-2-lax-1a.
4. Choose an instance type or add Amazon EBS storage volumes for uniform instance groups or instance fleets.
5. Choose any other options that apply to your cluster.
6. To launch your cluster, choose **Create cluster**.

CLI

To create a cluster on a Local Zone with 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 for use with Amazon EMR clusters

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 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 Amazon EMR 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 for Amazon EMR cluster cleanup.
- **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 an Amazon EMR cluster in the starting, running, or waiting states.

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 an Amazon EMR cluster to continue or terminate after step execution
- Using an auto-termination policy for Amazon EMR cluster cleanup
- Using termination protection to protect your Amazon EMR clusters from accidental shut down

Configuring an Amazon EMR 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 with 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 to protect your Amazon EMR clusters from accidental shut down.

> **Note**
>
> Amazon EMR automatically enables termination protection for all clusters with multiple primary 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. To shut down a cluster with multiple primary nodes, you must first modify the cluster attributes to disable termination

protection. For instructions, see Terminate an Amazon EMR Cluster with multiple primary nodes.

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 with 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 an Amazon EMR cluster.

Console

To turn on termination after step execution with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose **Create cluster**.
3. Under **Steps**, choose **Add step**. In the **Add step** dialog, enter appropriate field values. Options differ depending on the step type. To add your step and exit the dialog, choose **Add step**.
4. Under **Cluster termination**, select the **Terminate cluster after last step completes** check box.
5. Choose any other options that apply to your cluster.
6. To launch your cluster, choose **Create cluster**.

AWS CLI

To turn on termination after step execution with the AWS CLI

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

The following example demonstrates how to use 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-7.10.0 \
--applications Name=Hive Name=Pig --use-default-roles --ec2-attributes
 KeyName=myKey \
--steps Type=PIG,Name="Pig Program",ActionOnFailure=CONTINUE,\
Args=[-f,s3://amzn-s3-demo-bucket/scripts/pigscript.pig,-p,\
INPUT=s3://amzn-s3-demo-bucket/inputdata/,-p,OUTPUT=s3://amzn-s3-demo-bucket/
outputdata/,\
$INPUT=s3://amzn-s3-demo-bucket/inputdata/,$OUTPUT=s3://amzn-s3-demo-bucket/
outputdata/]
--instance-type m5.xlarge --instance-count 3 --auto-terminate
```

API

To turn off termination after step execution with the Amazon EMR API in cluster launch

1. When you use the RunJobFlow action to create a cluster, set the KeepJobFlowAliveWhenNoSteps property to `false`.
2. To change your configuration of termination after step execution with the Amazon EMR API post cluster launch:

   Use SetKeepJobFlowAliveWhenNoSteps action.

Using an auto-termination policy for Amazon EMR cluster cleanup

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 5.34.0 and later, and 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 • There are no pending steps
Amazon EMR versions 5.30.0 - 5.33.0 and 6.1.0 - 6.3.0	• There are no active YARN applications • 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 primary 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:

- In the following AWS Regions, Amazon EMR auto-termination is available with Amazon EMR 6.14.0 and higher:
  - Europe (Spain) (eu-south-2)
- In the following AWS Regions, Amazon EMR auto-termination is available with Amazon EMR 5.30.0 and 6.1.0 and higher:
  - US East (N. Virginia) (us-east-1)
  - US East (Ohio) (us-east-2)
  - US West (Oregon) (us-west-2)
  - US West (N. California) (us-west-1)
  - Africa (Cape Town) (af-south-1)
  - Asia Pacific (Hong Kong) (ap-east-1)
  - Asia Pacific (Mumbai) (ap-south-1)
  - Asia Pacific (Hyderabad) (ap-south-2)
  - Asia Pacific (Seoul) (ap-northeast-2)
  - Asia Pacific (Osaka) (ap-northeast-3)
  - Asia Pacific (Singapore) (ap-southeast-1)
  - Asia Pacific (Sydney) (ap-southeast-2)
  - Asia Pacific (Jakarta) (ap-southeast-3)
  - Asia Pacific (Tokyo) (ap-northeast-1)

  - Canada (Central) (ca-central-1)
  - South America (São Paulo) (sa-east-1)
  - Europe (Frankfurt) (eu-central-1)
  - Europe (Zurich) (eu-central-2)
  - Europe (Ireland) (eu-west-1)
  - Europe (London) (eu-west-2)
  - Europe (Milan) (eu-south-1)
  - Europe (Paris) (eu-west-3)
  - Europe (Stockholm) (eu-north-1)
  - Israel (Tel Aviv) (il-central-1)
  - Middle East (UAE) (me-central-1)
  - China (Beijing) (cn-north-1)
  - China (Ningxia) (cn-northwest-1)
  - AWS GovCloud (US-East) (us-gov-east-1)
  - AWS GovCloud (US-West) (us-gov-west-1)
- 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 with 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.
- 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.
- (EMR 5.30.0 and later) If you remove the default **Allow All** outbound rule to 0.0.0.0/ for the primary 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 primary security group. For more information about configuring security groups, see Amazon EMR-managed security group for the primary 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 the IAM resources that manage your EMR cluster.

```
{
  "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.

Console

To attach an auto-termination policy when you create a cluster with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose **Create cluster**.
3. Under **Cluster termination**, select **Terminate cluster after idle time**.
4. Specify the number of idle hours and minutes that can elapse before the cluster auto-terminates. The default idle time is 1 hour.
5. Choose any other options that apply to your cluster.
6. To launch your cluster, choose **Create cluster**.

To attach, update, or remove an auto-termination policy on a running cluster with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and select the cluster that you want to update.
3. On the **Properties** tab of the cluster details page, find **Cluster termination** and select **Edit**.
4. Select or clear **Enable auto-termination** to turn the feature on or off. If you turn on auto-termination, specify the number of idle hours and minutes that can elapse before the cluster auto-terminates. Then select **Save changes** to confirm.

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 with 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 to protect your Amazon EMR clusters from accidental shut down

Termination protection protects your clusters from accidental termination, which can be especially useful for long running clusters processing critical workloads. 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. 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 you use 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. This is true as well if you're running Amazon EMR releases 7.1 and higher and an instance becomes unhealthy and unrecoverable. 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 with the AWS Management Console or AWS CLI. For more information, see Manually resize a running Amazon EMR cluster.
- 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 in Amazon EMR.
- Removing instances from an instance fleet by reducing target capacity. For more information, see Instance fleet options.

Termination protection and Amazon EC2

The termination protection setting in an Amazon EMR cluster corresponds with the `DisableApiTermination` attribute for all Amazon EC2 instances in the cluster. For example, if you enable termination protection in an EMR cluster, Amazon EMR automatically sets `DisableApiTermination` to true for all EC2 instances within the EMR cluster. The same applies if you disable termination protection. Amazon EMR automatically sets `DisableApiTermination`

to false for all EC2 instances within the EMR cluster. If you terminate or scale down a cluster from Amazon EMR and the Amazon EC2 settings conflict for an EC2 instance, Amazon EMR prioritizes the Amazon EMR setting over the `DisableApiStop` and `DisableApiTermination` settings in Amazon EC2 and continues to terminate the EC2 instance.

For example, you can use the Amazon EC2 console to enable termination protection on an Amazon EC2 instance in an EMR cluster with termination protection disabled. If you terminate or scale down the cluster with the Amazon EMR console, the AWS CLI, or the Amazon EMR API, Amazon EMR overrides the `DisableApiTermination` setting, sets it to false, and terminates the instance along with other instances.

You can also use the Amazon EC2 console to enable stop protection on an Amazon EC2 instance in an EMR cluster with termination protection disabled. If you terminate or scale down the cluster, Amazon EMR sets `DisableApiStop` to false in Amazon EC2 and terminates the instance along with other instances.

Amazon EMR overrides the `DisableApiStop` setting only when you terminate or scale down a cluster. When you enable or disable termination protection in an EMR cluster, Amazon EMR doesn't change the `disableApiStop` setting for any of the EC2 instances in the respective EMR cluster.

> **Important**
>
> If you create an instance as part of an Amazon EMR cluster with termination protection, and you use the Amazon EC2 API or AWS CLI commands to modify the instance so that `DisableApiTermination` is `false`, and then the Amazon EC2 API or AWS CLI commands run the `TerminateInstances` operation, 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 adds the node to a denylist and does not allocate YARN containers to it until it becomes healthy again. Depending on the statuses of termination protection, unhealthy node replacement, and Amazon EMR release version, Amazon EMR will either replace the unhealthy instance or stop allocating controllers to the instance.

Termination protection and termination after step execution

When you enable termination after 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 an Amazon EMR cluster.

If a step fails that is configured with `ActionOnFailure` set to `CANCEL_AND_WAIT`, if termination after 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	Termination after 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.

For single-node clusters, default termination protection settings are as follows:

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

For high-availability clusters, default termination protection settings are as follows:

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

Console

To turn termination protection on or off when you create a cluster with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose **Create cluster**.
3. For **EMR release version**, choose **emr-6.6.0** or later.
4. Under **Cluster termination and node replacement**, make sure that **Use termination protection** is pre-selected, or clear the selection to turn it off.
5. Choose any other options that apply to your cluster.
6. To launch your cluster, choose **Create cluster**.

AWS CLI

To turn termination protection on or off when you create a cluster using the AWS CLI

- With the AWS CLI, you can launch a cluster with termination protection enabled with 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-7.10.0 \
  --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 with the console or the AWS CLI.

Console

To turn termination protection on or off for a running cluster with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and select the cluster that you want to update.
3. On the **Properties** tab on the cluster details page, find **Cluster termination** and select **Edit**.

4. Select or clear the **Use termination protection** check box to turn the feature on or off. Then select **Save changes** to confirm.

AWS CLI

To turn termination protection on or off for a running cluster using the AWS CLI

- To enable termination protection on a running cluster with 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
  ```

Replacing unhealthy nodes with Amazon EMR

Amazon EMR periodically uses the NodeManager health checker service in Apache Hadoop to monitor the statuses of core nodes in your Amazon EMR on Amazon EC2 clusters. If a node is not functioning optimally, the node is marked as unhealthy and the health checker reports that node to the Amazon EMR controller. The Amazon EMR controller adds the node to a deny list, preventing the node from receiving new YARN applications until the status of the node improves.

> **Note**
>
> A common reason for a node to be unhealthy is that it is out of disk space. For more information about when a core node is almost out of disk space, the following **re:Post Knowledge Center** article is helpful: Why is the core node in my Amazon EMR cluster running out of disk space?

> **Note**
>
> Hadoop does provide the ability to run customized node-health checks. This is explained in further detail in the Apache Hadoop documentation at NodeManager.

You can choose whether Amazon EMR should terminate unhealthy nodes or keep them in the cluster. If you turn off unhealthy-node replacement, they stay in the deny list and continue to count toward cluster capacity. You can still connect to your Amazon EC2 core instance for configuration and recovery, so you can resize your cluster if you want to add capacity. For more information about how node replacement and termination work, see Using termination protection.

If unhealthy node replacement is turned on, Amazon EMR terminates an unhealthy core node and provisions a new instance, based on the number of instances in the instance group, or based on the target capacity for instance fleets. If any nodes are unhealthy for more than 45 minutes, Amazon EMR will gracefully replace the nodes. If graceful decommissioning for a node doesn't complete within one hour, the node is forcefully terminated, unless terminating it brings the cluster below replication factor or HDFS capacity constraints.

> **Important**
>
> Note that the time it takes before a node is gracefully decommissioned or terminated can be subject to change.
> While unhealthy node replacement significantly mitigates the possibility for data loss, it doesn't eliminate the risk entirely. HDFS data can be permanently lost during the graceful replacement of an unhealthy core instance. We recommend that you always back up your data.

For more information about identifying unhealthy nodes and recovery, see Resource errors. Additionally, for more best practices you can follow in order to maintain the health of a cluster, see the following documentation for the resource error Amazon EMR cluster terminates with NO_SLAVE_LEFT and core nodes FAILED_BY_MASTER.

Amazon EMR publishes Amazon CloudWatch Events for unhealthy node replacement, so you can keep track of what's happening with your unhealthy core instances. For more information, see unhealthy node replacement events.

Default node replacement and termination protection settings

Unhealthy node replacement is available for all Amazon EMR releases, but the default settings depend on the release label you choose. You can change any of these settings by configuring unhealthy node replacement when creating a new cluster or by going to cluster configuration at any time.

If you're creating a single-node cluster or high-availability cluster that is running Amazon EMR release 7.0 or lower, the default setting of unhealthy node replacement is dependent on termination protection:

- Enabling termination protection **disables** unhealthy node replacement.
- Disabling termination protection **enables** unhealthy node replacement.

Configuring unhealthy node replacement when you launch a cluster

You can enable or disable unhealthy node replacement when you launch a cluster using the console, the AWS CLI, or the API.

The default unhealthy node replacement setting depends on how you launch the cluster:

- Amazon EMR console — unhealthy node replacement is **enabled** by default.
- AWS CLI `aws emr create-cluster` — unhealthy node replacement is **enabled** by default unless you specify `--no-unhealthy-node-replacement`.
- Amazon EMR RunJobFlow API command — unhealthy node replacement is **enabled** by default unless you set the `UnhealthyNodeReplacement` Boolean value to `True` or `False`.

Console

To turn unhealthy node replacement on or off when you create a cluster with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose **Create cluster**.
3. For **EMR release version**, choose the Amazon EMR release label you want.

4. Under **Cluster termination and node replacement**, make sure that **Unhealthy node replacement (recommended)** is pre-selected, or clear the selection to turn it off.
5. Choose any other options that apply to your cluster.
6. To launch your cluster, choose **Create cluster**.

AWS CLI

To turn unhealthy node replacement on or off when you create a cluster using the AWS CLI

- With the AWS CLI, you can launch a cluster with unhealthy node replacement enabled with the `create-cluster` command with the `--unhealthy-node-replacement` parameter. Unhealthy node replacement is on by default.

  The following example creates a cluster with unhealthy node replacement 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 "SampleCluster" --release-label emr-7.10.0 \
  --applications Name=Hadoop Name=Hive Name=Pig \
  --use-default-roles --ec2-attributes KeyName=myKey --instance-type m5.xlarge \
  --instance-count 3 --unhealthy-node-replacement
  ```

  For more information about using Amazon EMR commands in the AWS CLI, see Amazon EMR AWS CLI commands.

Configuring unhealthy node replacement in a running cluster

You can turn unhealthy node replacement on or off for a running cluster using the console, the AWS CLI, or the API.

Console

To turn unhealthy node replacement on or off for a running cluster with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and select the cluster that you want to update.
3. On the **Properties** tab on the cluster details page, find **Cluster termination and node replacement** and select **Edit**.
4. Select or clear the **unhealthy node replacement** check box to turn the feature on or off. Then select **Save changes** to confirm.

AWS CLI

To turn unhealthy node replacement on or off for a running cluster using the AWS CLI

- To turn on unhealthy node replacement on a running cluster with the AWS CLI, use the `modify-cluster-attributes` command with the `--unhealthy-node-replacement` parameter. To disable it, use the `--no-unhealthy-node-replacement` parameter.

  The following example turns on unhealthy node replacement on the cluster with ID *`j-3KVTXXXXXX7UG`*:

  ```
  aws emr modify-cluster-attributes --cluster-id j-3KVTXXXXXX7UG --unhealthy-node-
  replacement
  ```

  The following example turns off unhealthy node replacement on the same cluster:

  ```
  aws emr modify-cluster-attributes --cluster-id j-3KVTXXXXXX7UG --no-unhealthy-
  node-replacement
  ```

Working with Amazon Linux AMIs in Amazon EMR

Amazon Linux Amazon Machine Images (AMIs)

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 (AL) 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. When you use Amazon EMR 5.7.0 or higher, 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 to provide more flexibility for Amazon EMR cluster configuration.

Amazon EMR automatically attaches an Amazon EBS General Purpose SSD volume as the root device for all AMIs. EBS-backed AMIs enhance performance. 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 options and behavior in Amazon EMR.

Topics

- Using the default Amazon Linux AMI for Amazon EMR
- Using a custom AMI to provide more flexibility for Amazon EMR cluster configuration
- Change the Amazon Linux release when you create an EMR cluster
- Customizing the Amazon EBS root device volume

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. Starting with Amazon EMR 5.36, Amazon EMR 6.6, and Amazon EMR 7.0 releases the default behavior for updating Amazon Linux 2 (AL2 for EMR 5.x and 6.x, AL2023 for

EMR 7.x) in an Amazon EMR default AMI is to automatically apply the latest Amazon Linux release for the default Amazon EMR AMI.

Automatic Amazon Linux updates for Amazon EMR releases

When you launch a cluster with *the latest patch release* of Amazon EMR 7.0 or higher, 6.6 or higher, or 5.36 or higher, Amazon EMR uses the latest Amazon Linux release for the default Amazon EMR AMI. For example:

- Where there is an `x.x.0` and an `x.x.1` release, the `x.x.0` release stops getting AMI updates when `x.x.1` launches.
- Similarly, `x.x.1` stops getting AMI updates when `x.x.2` launches.
- Later, when `x.y.0` releases, `x.x.[latest]` continues to receive AMI updates alongside `x.y.[latest]`.

To see if you're using the latest patch release as denoted by the number after the second decimal point (`6.8.1`) for an Amazon EMR release, refer to the available releases in the *Amazon EMR Release Guide*, check the **Amazon EMR release** dropdown when you create a cluster in the console, or use the `ListReleaseLabels` API or `list-release-labels` CLI action. To get updates when we launch a new Amazon EMR release, subscribe to the RSS feed on the What's new? page in the Release Guide.

If you want to, you can choose to launch your cluster with the Amazon Linux version that the Amazon EMR release first shipped with. For information on how to specify the Amazon Linux release for your cluster, see Change the Amazon Linux release when you create an EMR cluster.

Default Amazon Linux versions

Topics

- Default AMIs for Amazon EMR 7.0 and higher
- Default AMIs for Amazon EMR 6.6 and higher
- Default AMIs for Amazon EMR 5.x

Default AMIs for Amazon EMR 7.0 and higher

The following table lists Amazon Linux information for the latest patch version of Amazon EMR releases 7.0 and higher.

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.8.2 250721.	6.1.144-170.251.am zn2023	August 14, 2025	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-central-1 • eu-west-1 • eu-west-2 • eu-south-1 • eu-west-3 • eu-south-2 • eu-north-1 • eu-central-2 • ap-east-1 • ap-south-2 • ap-southeast-3 • ap-southeast-5 • ap-southeast-4 • ap-south-1 • ap-northeast-3 • ap-northeast-2 • ap-southeast-1 • ap-southeast-2 • ap-southeast-7 (7.1.0+) • ap-northeast-1 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-south-1 • me-central-1 • ca-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northwest-1 • il-central-1 • mx-central-1 (7.1.0+)

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.7.2 250623.	6.1.141-155.222.am zn2023	July 21, 2025	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-central-1 • eu-west-1 • eu-west-2 • eu-south-1 • eu-west-3 • eu-south-2 • eu-north-1 • eu-central-2 • ap-east-1 • ap-south-2 • ap-southeast-3 • ap-southeast-5 • ap-southeast-4 • ap-south-1 • ap-northeast-3 • ap-northeast-2 • ap-southeast-1 • ap-southeast-2 • ap-southeast-7 (7.1.0+) • ap-northeast-1 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-south-1 • me-central-1 • ca-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northwest-1 • il-central-1 • mx-central-1 (7.1.0+)

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.7.2 250609.(	6.1.140-154.222.am zn2023	July 14, 2025	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-central-1 • eu-west-1 • eu-west-2 • eu-south-1 • eu-west-3 • eu-south-2 • eu-north-1 • eu-central-2 • ap-east-1 • ap-south-2 • ap-southeast-3 • ap-southeast-5 • ap-southeast-4 • ap-south-1 • ap-northeast-3 • ap-northeast-2 • ap-southeast-1 • ap-southeast-2 • ap-southeast-7 (7.1.0+) • ap-northeast-1 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-south-1 • me-central-1 • ca-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northwest-1 • il-central-1 • mx-central-1 (7.1.0+)

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.7.2 250527.	6.1.134-152.225.am zn2023	Jun 19, 2025	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-central-1 • eu-west-1 • eu-west-2 • eu-south-1 • eu-west-3 • eu-south-2 • eu-north-1 • eu-central-2 • ap-east-1 • ap-south-2 • ap-southeast-3 • ap-southeast-5 • ap-southeast-4 • ap-south-1 • ap-northeast-3 • ap-northeast-2 • ap-southeast-1 • ap-southeast-2 • ap-southeast-7 (7.1.0+) • ap-northeast-1 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-south-1 • me-central-1 • ca-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northwest-1 • il-central-1 • mx-central-1 (7.1.0+)

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.7.2 250512.(	6.1.134-152.225.am zn2023	Jun 04, 2025	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-central-1 • eu-west-1 • eu-west-2 • eu-south-1 • eu-west-3 • eu-south-2 • eu-north-1 • eu-central-2 • ap-east-1 • ap-south-2 • ap-southeast-3 • ap-southeast-5 • ap-southeast-4 • ap-south-1 • ap-northeast-3 • ap-northeast-2 • ap-southeast-1 • ap-southeast-2 • ap-southeast-7 (7.1.0+) • ap-northeast-1 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-south-1 • me-central-1 • ca-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northwest-1 • il-central-1 • mx-central-1 (7.1.0+)

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.7.2 250428.	6.1.134-150.224.am zn2023	May 23, 2025	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-central-1 • eu-west-1 • eu-west-2 • eu-south-1 • eu-west-3 • eu-south-2 • eu-north-1 • eu-central-2 • ap-east-1 • ap-south-2 • ap-southeast-3 • ap-southeast-5 • ap-southeast-4 • ap-south-1 • ap-northeast-3 • ap-northeast-2 • ap-southeast-1 • ap-southeast-2 • ap-southeast-7 (7.1.0+) • ap-northeast-1 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-south-1 • me-central-1 • ca-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northwest-1 • il-central-1 • mx-central-1 (7.1.0+)

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.7.2 250414.(	6.1.132-147.221.am zn2023	May 12, 2025	• 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-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• il-central-1 • ca-west-1 • ap-southeast-7 • ap-southeast-5 • mx-central-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.7.2 250331.	6.1.131-143.221.am zn2023	April 18, 2025	• 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-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• il-central-1 • ca-west-1 • ap-southeast-7 • ap-southeast-5 • mx-central-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.6.2 250303.(	6.1.129-138.220.am zn2023	March 27, 2025	• 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-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• il-central-1 • ca-west-1 • ap-southeast-7 • ap-southeast-5 • mx-central-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.6.2 250218.:	6.1.128-136.201.am zn2023	March 08, 2025	• 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-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• il-central-1 • ca-west-1 • ap-southeast-7 • ap-southeast-5 • mx-central-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.6.2 250211.(	6.1.127-135.201.am zn2023	February 26, 2025	• 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-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• il-central-1 • ca-west-1 • ap-southeast-7 • ap-southeast-5 • mx-central-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.6.2 250123.	6.1.124-134.200.am zn2023	January 27, 2025	• 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-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• il-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.6.2 250115.(	6.1.119-129.201.am zn2023	January 23, 2025	• 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-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• il-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.6.2 241121.	6.1.115-126.197.am zn2023	December 12, 2024	• 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-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• il-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.5.2 241031.(	6.1.112-124.190.am zn2023	November 15, 2024	• 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-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• il-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.5.2 241001.	6.1.109-118.189.am zn2023	October 4, 2024	• 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-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• il-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.5.2 240819.(	6.1.102-111.182.am zn2023	August 20, 2024	• 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-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• il-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.5.2 240730.	6.1.97-104.177.amzn2023	August 2, 2024	• 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-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• il-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.5.2 240722.(	6.1.97-104.177.amzn2023	July 24, 2024	• 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-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• il-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.5.2 240708.	6.1.96-102.177.amzn2023	July 23, 2024	• 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-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• il-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.3.2 240304.(	6.1.79-99.164.amzn2023	March 12, 2024	• 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-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• il-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.3.2 240219.	6.1.77-99.164.amzn2023	March 1, 2024	• 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-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• il-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.3.2 240205.(	6.1.75-99.163.amzn2023	February 19, 2024	• 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-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• il-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.3.2 240122.	6.1.72-96.166.amzn2023	February 5, 2024	• 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-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• il-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.3.2 240108.(	6.1.72-96.166.amzn2023	January 24, 2024	• 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-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• il-central-1 • ca-west-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2023.3.2 231211.	6.1.66-91.160.amzn2023	December 19, 2023	• 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-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• il-central-1 • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

Default AMIs for Amazon EMR 6.6 and higher

The following table lists Amazon Linux information for the latest patch version of Amazon EMR releases 6.6.x and higher.

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2025 610.0	4.14.355-277.647.amzn2	July 14, 2025	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-central-1 • eu-west-1 • eu-west-2 • eu-south-1 • eu-west-3 • eu-south-2 (6.7.0+) • eu-north-1 • eu-central-2 (6.7.0+) • ap-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• ap-south-2 (6.7.0+) • ap-southeast-3 • ap-southeast-5 (6.11.1 and 6.14.0+) • ap-southeast-4 (6.8.0+) • ap-south-1 • ap-northeast-3 • ap-northeast-2 • ap-southeast-1 • ap-southeast-2 • ap-southeast-7 (6.11.1 and 6.14.0+) • ap-northeast-1 • af-south-1 • sa-east-1 • me-south-1 • me-central-1 (6.7.0+) • ca-central-1 • ca-west-1 (6.9.0+) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northwest-1 • il-central-1 (6.7.0+)

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• `mx-central-1` (6.11.1 and 6.14.0+)

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2025 527.1	4.14.355-277.647.amzn2	Jun 19, 2025	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-central-1 • eu-west-1 • eu-west-2 • eu-south-1 • eu-west-3 • eu-south-2 (6.7.0+) • eu-north-1 • eu-central-2 (6.7.0+) • ap-east-1 • ap-south-2 (6.7.0+) • ap-southeast-3 • ap-southeast-5 (6.11.1 and 6.14.0+) • ap-southeast-4 (6.8.0+) • ap-south-1 • ap-northeast-3 • ap-northeast-2 • ap-southeast-1 • ap-southeast-2 • ap-southeast-7 (6.11.1 and 6.14.0+) • ap-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• af-south-1 • sa-east-1 • me-south-1 • me-central-1 (6.7.0+) • ca-central-1 • ca-west-1 (6.9.0+) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northwest-1 • il-central-1 (6.7.0+) • mx-central-1 (6.11.1 and 6.14.0+)

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2025 512.0	4.14.355-277.643.amzn2	Jun 04, 2025	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-central-1 • eu-west-1 • eu-west-2 • eu-south-1 • eu-west-3 • eu-south-2 (6.7.0+) • eu-north-1 • eu-central-2 (6.7.0+) • ap-east-1 • ap-south-2 (6.7.0+) • ap-southeast-3 • ap-southeast-5 (6.11.1 and 6.14.0+) • ap-southeast-4 (6.8.0+) • ap-south-1 • ap-northeast-3 • ap-northeast-2 • ap-southeast-1 • ap-southeast-2 • ap-southeast-7 (6.11.1 and 6.14.0+) • ap-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• af-south-1 • sa-east-1 • me-south-1 • me-central-1 (6.7.0+) • ca-central-1 • ca-west-1 (6.9.0+) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northwest-1 • il-central-1 (6.7.0+) • mx-central-1 (6.11.1 and 6.14.0+)

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2025 428.0	4.14.355-276.639.amzn2	May 23, 2025	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-central-1 • eu-west-1 • eu-west-2 • eu-south-1 • eu-west-3 • eu-south-2 (6.7.0+) • eu-north-1 • eu-central-2 (6.7.0+) • ap-east-1 • ap-south-2 (6.7.0+) • ap-southeast-3 • ap-southeast-5 (6.11.1 and 6.14.0+) • ap-southeast-4 (6.8.0+) • ap-south-1 • ap-northeast-3 • ap-northeast-2 • ap-southeast-1 • ap-southeast-2 • ap-southeast-7 (6.11.1 and 6.14.0+) • ap-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• af-south-1 • sa-east-1 • me-south-1 • me-central-1 (6.7.0+) • ca-central-1 • ca-west-1 (6.9.0+) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northwest-1 • il-central-1 (6.7.0+) • mx-central-1 (6.11.1 and 6.14.0+)

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2025 414.0	4.14.355-276.618.amzn2	May 12, 2025	• 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-central-2 • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• `il-central-1` • `ca-west-1` (6.9.0+ and 5.36.1+) • `us-gov-east-1` • `us-gov-west-1` • `cn-north-1` • `cn-northeast-1`

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2025 321.0	4.14.355	April 09, 2025	• 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-central-2 • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• `il-central-1` • `ca-west-1` (6.9.0+ and 5.36.1+) • `us-gov-east-1` • `us-gov-west-1` • `cn-north-1` • `cn-northeast-1`

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2025 305.0	4.14.355	March 18, 2025	• 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-central-2 • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• `il-central-1` • `ca-west-1` (6.9.0+ and 5.36.1+) • `us-gov-east-1` • `us-gov-west-1` • `cn-north-1` • `cn-northeast-1`

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2025 220.0	4.14.355	March 08, 2025	• 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-central-2 • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• `il-central-1` • `ca-west-1` (6.9.0+ and 5.36.1+) • `us-gov-east-1` • `us-gov-west-1` • `cn-north-1` • `cn-northeast-1`

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2025 201.0	4.14.355	February 28, 2025	• 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-central-2 • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• `il-central-1` • `ca-west-1` (6.9.0+ and 5.36.1+) • `us-gov-east-1` • `us-gov-west-1` • `cn-north-1` • `cn-northeast-1`

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2025 123.4	4.14.355	January 27, 2025	• 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-central-2 • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• `il-central-1` • `ca-west-1` (6.9.0+ and 5.36.1+) • `us-gov-east-1` • `us-gov-west-1` • `cn-north-1` • `cn-northeast-1`

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2025 116.0	4.14.355	January 23, 2025	• 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-central-2 • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• `il-central-1` • `ca-west-1` (6.9.0+ and 5.36.1+) • `us-gov-east-1` • `us-gov-west-1` • `cn-north-1` • `cn-northeast-1`

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2024 217.0	4.14.355	January 8, 2025	• 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-central-2 • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• il-central-1 • ca-west-1 (6.9.0+ and 5.36.1+) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2024 001.0	4.14.352	October 4, 2024	• 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-central-2 (6.10.1+) • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 (6.8.1+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10.1+) • me-south-1 • ca-central-1 • il-central-1 (6.8.1+ and 5.36.1) • ca-west-1 (6.9.1+ and 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2024 816.0	4.14.350	August 21, 2024	• 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-central-2 (6.10.1+) • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 (6.8.1+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10.1+) • me-south-1 • ca-central-1 • il-central-1 (6.8.1+ and 5.36.1) • ca-west-1 (6.9.1+ and 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2024 809.0	4.14.349	August 20, 2024	• 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-central-2 (6.10.1+) • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 (6.8.1+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10.1+) • me-south-1 • ca-central-1 • il-central-1 (6.8.1+ and 5.36.1) • ca-west-1 (6.9.1+ and 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2024 719.0	4.14.348	July 25, 2024	• 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-central-2 (6.10.1+) • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 (6.8.1+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10.1+) • me-south-1 • ca-central-1 • il-central-1 (6.8.1+ and 5.36.1) • ca-west-1 (6.9.1+ and 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2024 709.1	4.14.348	July 23, 2024	• 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-central-2 (6.10.1+) • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 (6.8.1+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10.1+) • me-south-1 • ca-central-1 • il-central-1 (6.8.1+ and 5.36.1) • ca-west-1 (6.9.1+ and 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2024 223.0	4.14.336	March 8, 2024	• 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-central-2 (6.10.1+) • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 (6.8.1+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10.1+) • me-south-1 • ca-central-1 • il-central-1 (6.8.1+ and 5.36.1) • ca-west-1 (6.9.1+ and 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2024 131.0	4.14.336	February 14, 2024	• 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-central-2 (6.10.1+) • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 (6.8.1+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10.1+) • me-south-1 • ca-central-1 • il-central-1 (6.8.1+ and 5.36.1) • ca-west-1 (6.9.1+ and 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2024 124.0	4.14.336	February 7, 2024	• 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-central-2 (6.10.1+) • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 (6.8.1+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10.1+) • me-south-1 • ca-central-1 • il-central-1 (6.8.1+ and 5.36.1) • ca-west-1 (6.9.1+ and 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2024 109.0	4.14.334	January 24, 2024	• 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-central-2 (6.10.1+) • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 (6.8.1+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10.1+) • me-south-1 • ca-central-1 • il-central-1 (6.8.1+ and 5.36.1) • ca-west-1 (6.9.1+ and 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 218.0	4.14.330	January 2, 2024	• 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-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-southeast-4 (6.8+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10+) • me-south-1 • ca-central-1 • il-central-1 (6.8+ and 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 206.0	4.14.330	December 22, 2023	• 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-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-southeast-4 (6.8+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10+) • me-south-1 • ca-central-1 • il-central-1 (6.8+ and 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 116.0	4.14.328	December 11, 2023	• 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-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-southeast-4 (6.8+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10+) • me-south-1 • ca-central-1 • il-central-1 (6.8+ and 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 101.0	4.14.327	November 17, 2023	• 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-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-southeast-4 (6.8+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10+) • me-south-1 • ca-central-1 • il-central-1 (6.8+ and 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 020.1	4.14.326	November 07, 2023	• 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-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-southeast-4 (6.8+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10+) • me-south-1 • ca-central-1 • il-central-1 (6.8+ and 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 012.1	4.14.326	October 26, 2023	• 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-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-southeast-4 (6.8+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10+) • me-south-1 • ca-central-1 • il-central-1 (6.8+ and 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 926.0	4.14.322	October 19, 2023	• 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-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-southeast-4 (6.8+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10+) • me-south-1 • ca-central-1 • il-central-1 (6.8+ and 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 8906.0	4.14.322	October 04, 2023	• 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-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-southeast-4 (6.8+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10+) • me-south-1 • ca-central-1 • il-central-1 (6.9+ and 5.36.1)

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 822.0	4.14.322	August 30, 2023	• 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-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-southeast-4 (6.8+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10+) • me-south-1 • ca-central-1 • il-central-1 (6.9+ and 5.36.1)

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 808.0	4.14.320	August 24, 2023	• 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-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-southeast-4 (6.8+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10+) • me-south-1 • ca-central-1 • il-central-1 (6.9+ and 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 727.0	4.14.320	August 14, 2023	• 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-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-southeast-4 (6.8+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10+) • me-south-1 • ca-central-1 • il-central-1 (6.9+ and 5.36.1)

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 719.0	4.14.320	August 2, 2023	• 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-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • ap-southeast-3 • ap-southeast-4 (6.8+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10+) • me-south-1 • ca-central-1 • il-central-1 (6.9+ and 5.36.1)

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 628.0	4.14.318	July 12, 2023	• 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-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • 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-central-1 (6.10+) • me-south-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 612.0	4.14.314	June 23, 2023	• 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-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • 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-central-1 (6.10+) • me-south-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• `ca-central-1`

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 504.1	4.14.313	May 16, 2023	• 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-central-2 (6.10+) • eu-south-1 • eu-south-2 (6.10+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10+) • 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-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 418.0	4.14.311	May 3, 2023	• 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-central-2 (6.10 only) • eu-south-1 • eu-south-2 (6.10 only) • ap-east-1 • ap-south-1 • ap-south-2 (6.10 only) • 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-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 404.1	4.14.311	April 18, 2023	• 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-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • 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-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 404.0	4.14.311	April 10, 2023	• us-east-1 • eu-west-3

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 320.0	4.14.309	March 30, 2023	• 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-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • 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-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 307.0	4.14.305	March 15, 2023	• 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-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • 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-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 207.0	4.14.304	March 3, 2023	• 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-central-2 • eu-south-1 • eu-south-2 • ap-east-1 • ap-south-1 • ap-south-2 • 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-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 119.1	4.14.301	February 9, 2023	• 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

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2022 210.1	4.14.301	January 12, 2023	• 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

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2022 103.3	4.14.296	December 5, 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

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2022 004.0	4.14.294	November 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

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2022 912.1	4.14.291	October 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
2.0.2022 805.0	4.14.287	August 30, 2022	• us-west-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2022 719.0	4.14.287	August 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

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2022 426.0	4.14.281	June 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

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2022 406.1	4.14.275	May 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

Default AMIs for Amazon EMR 5.x

The following table lists Amazon Linux information for the latest patch version of Amazon EMR 5.x releases 5.36 and higher.

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2025 428.0	4.14.355-276.639.amzn2	May 23, 2025	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • eu-central-1 • eu-west-1 • eu-west-2 • eu-south-1 • eu-west-3 • eu-south-2 • eu-north-1 • eu-central-2 • ap-east-1 • ap-south-2 • ap-southeast-3 • ap-southeast-5 (5.36.2) • ap-southeast-4 • ap-south-1 • ap-northeast-3 • ap-northeast-2 • ap-southeast-1 • ap-southeast-2 • ap-southeast-7 (5.36.2) • ap-northeast-1 • af-south-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• sa-east-1 • me-south-1 • me-central-1 • ca-central-1 • ca-west-1 (5.36.1+) • us-gov-east-1 (5.36.1+) • us-gov-west-1 (5.36.1+) • cn-north-1 (5.36.1+) • cn-northwest-1 (5.36.1+) • il-central-1 • mx-central-1 (5.36.2)

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2025 321.0	4.14.355	April 09, 2025	• 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-central-2 • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• `il-central-1` • `ca-west-1` (6.9.0+ and 5.36.1+) • `us-gov-east-1` • `us-gov-west-1` • `cn-north-1` • `cn-northeast-1`

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2025 305.0	4.14.355	March 18, 2025	• 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-central-2 • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• `il-central-1` • `ca-west-1` (6.9.0+ and 5.36.1+) • `us-gov-east-1` • `us-gov-west-1` • `cn-north-1` • `cn-northeast-1`

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2025 220.0	4.14.355	March 08, 2025	• 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-central-2 • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• `il-central-1` • `ca-west-1` (6.9.0+ and 5.36.1+) • `us-gov-east-1` • `us-gov-west-1` • `cn-north-1` • `cn-northeast-1`

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2025 201.0	4.14.355	February 28, 2025	• 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-central-2 • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• `il-central-1` • `ca-west-1` (6.9.0+ and 5.36.1+) • `us-gov-east-1` • `us-gov-west-1` • `cn-north-1` • `cn-northeast-1`

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2025 123.4	4.14.355	January 27, 2025	• 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-central-2 • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• il-central-1 • ca-west-1 (6.9.0+ and 5.36.1+) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2025 116.0	4.14.355	January 23, 2025	• 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-central-2 • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• `il-central-1` • `ca-west-1` (6.9.0+ and 5.36.1+) • `us-gov-east-1` • `us-gov-west-1` • `cn-north-1` • `cn-northeast-1`

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2024 217.0	4.14.355	January 8, 2025	• 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-central-2 • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1 • me-central-1 • me-south-1 • ca-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• `il-central-1` • `ca-west-1` (6.9.0+ and 5.36.1+) • `us-gov-east-1` • `us-gov-west-1` • `cn-north-1` • `cn-northeast-1`

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2024 816.0	4.14.350	August 21, 2024	• 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-central-2 (6.10.1+) • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 (6.8.1+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10.1+) • me-south-1 • ca-central-1 • il-central-1 (6.8.1+ and 5.36.1) • ca-west-1 (6.9.1+ and 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2024 809.0	4.14.349	August 20, 2024	• 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-central-2 (6.10.1+) • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 (6.8.1+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10.1+) • me-south-1 • ca-central-1 • il-central-1 (6.8.1+ and 5.36.1) • ca-west-1 (6.9.1+ and 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2024 719.0	4.14.348	July 25, 2024	• 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-central-2 (6.10.1+) • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 (6.8.1+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10.1+) • me-south-1 • ca-central-1 • il-central-1 (6.8.1+ and 5.36.1) • ca-west-1 (6.9.1+ and 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2024 709.1	4.14.348	July 23, 2024	• 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-central-2 (6.10.1+) • eu-south-1 • eu-south-2 (6.10.1+) • ap-east-1 • ap-south-1 • ap-south-2 (6.10.1+) • ap-southeast-3 • ap-southeast-4 (6.8.1+ and 5.36.1) • ap-northeast-1 • ap-northeast-2 • ap-northeast-3 • ap-southeast-1 • ap-southeast-2 • af-south-1 • sa-east-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
			• me-central-1 (6.10.1+) • me-south-1 • ca-central-1 • il-central-1 (6.8.1+ and 5.36.1) • ca-west-1 (6.9.1+ and 5.36.1) • us-gov-east-1 • us-gov-west-1 • cn-north-1 • cn-northeast-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 504.1	4.14.313	May 16, 2023	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • ca-central-1 • 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 • me-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 418.0	4.14.311	May 3, 2023	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • ca-central-1 • 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 • me-central-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 404.1	4.14.311	April 18, 2023	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • ca-central-1 • 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
2.0.2023 404.0	4.14.311	April 10, 2023	• us-east-1 • eu-west-3

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 320.0	4.14.309	March 30, 2023	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • ca-central-1 • 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

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 307.0	4.14.305	March 15, 2023	• us-east-1 • us-east-2 • us-west-1 • us-west-2 • ca-central-1 • 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

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2023 207.0	4.14.304	March 3, 2023	• 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

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2022 210.1	4.14.301	January 12, 2023	• 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

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2022 103.3	4.14.296	December 5, 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

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2022 004.0	4.14.294	November 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

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2022 912.1	4.14.291	October 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

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2022 719.0	4.14.287	August 10, 2022	• us-west-1 • eu-west-3 • eu-north-1 • eu-central-1 • ap-south-1 • me-south-1

OsRelea Label (AL versi	AL kernel version	Available date	AWS Regions
2.0.2022 426.0	4.14.281	June 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

Software update considerations

Take note of the following default software update behaviors:

Amazon EMR 7.x — Amazon Linux 2023

Amazon EMR releases 7.0 and higher run on Amazon Linux 2023 (AL2023). Your clusters only contain security updates that were available in the version of AL2023 AMI that you chose when you created them. At launch time, EMR cluster instances will not install the latest security updates from the enabled package repositories. To receive the latest security updates, we recommend that you periodically recreate your cluster. For more information on AL2023, see Updating Amazon Linux 2023 in the *Amazon Linux 2023 User Guide*.

Amazon EMR 5.36+ and 6.6+ — Amazon Linux 2

Amazon EMR releases 5.36 and higher, and 6.6 and higher, run on Amazon Linux 2 (AL2). Your clusters only contain security updates that were available in the version of AL2 AMI that you chose when you created them. At launch time, EMR cluster instances will not install the latest security updates from the enabled package repositories. To receive the latest security updates, we recommend that you periodically recreate your cluster. For more information on AL2, see Manage software on your AL2 instance in the *Amazon Linux 2 User Guide*.

Amazon EMR 3.x, 4.x, 5.0.0 to 5.35.0, and 6.0.0 to 6.5.0 — Amazon Linux AMI locked to Amazon EMR release version

For Amazon EMR releases 3.x, 4.x, 5.0.0 to 5.35.0, and 6.0.0 to 6.5.0, 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 in that release version.

When an Amazon EC2 instance boots for the first time in a cluster that is based on the default Amazon Linux (AL) or AL2 AMI for Amazon EMR, it checks for software updates that apply to the release version in the enabled package repositories for AL and Amazon EMR. As with other Amazon EC2 instances that run AL or AL2 AMIs, critical and important security updates from these repositories are automatically installed.

Also note that, in your networking configuration, you must allow HTTP and HTTPS egress to Amazon Linux repositories in Amazon S3. Otherwise, security updates will fail. For more information, see Amazon Linux - Package repository in the *Amazon EC2 User Guide*. 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 releases 3.x, 4.x, 5.0.0 to 5.35.0, and 6.0.0 to 6.5.0 use the 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 that run the default AL or AL2 AMIs. If new Amazon Linux software updates that require a reboot (such as kernel, NVIDIA, and CUDA updates) become available after an Amazon EMR release becomes available, EMR cluster instances that run the default AMI do not automatically download and install those updates. To get kernel updates, you can use the latest Amazon Linux AMI as described in Using a custom AMI to provide more flexibility for Amazon EMR cluster configuration.

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 with SSH or when you use a bootstrap action. This might cause incompatibilities because all packages are installed indiscriminately.

Software update best practices

Best practices for managing software updates

- If you use a lower release version of Amazon EMR, consider and test a migration to the latest release before updating software packages.
- If you migrate to a higher release version or you upgrade software packages, test the implementation in a non-production environment first. The option to clone clusters with 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 with an Amazon EMR cluster.

Using a custom AMI to provide more flexibility for Amazon EMR cluster configuration

When you use Amazon EMR 5.7.0 or higher, 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.
- 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 lower than 5.24.0. As with the default AMI, the minimum root volume size for a custom AMI is 10 GiB for Amazon EMR releases 6.9 and lower, and 15 GiB for Amazon EMR releases 6.10 and higher. For more information, see Creating a custom AMI with an encrypted Amazon EBS root device volume.

> **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.

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*.

> **Important**
>
> EMR clusters that run Amazon Linux or Amazon Linux 2 Amazon Machine Images (AMIs) 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 that run 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 release becomes available, EMR cluster instances that run 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.

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*.
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 and Creating a custom AMI with an encrypted Amazon EBS root device volume.

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)

Amazon EMR console

To specify a single custom AMI from the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose **Create cluster**.
3. Under **Name and applications**, find **Operating system options**. Choose **Custom AMI**, and enter your AMI ID in the **Custom AMI** field.
4. Choose any other options that apply to your cluster.
5. To launch your cluster, choose **Create cluster**.

AWS CLI

To specify a single custom AMI with 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 Customizing the Amazon EBS root device volume.

> **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 higher
- 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 higher, 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 (primary, 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 primary, 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 primary 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=PRIMARY,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 higher 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 higher, 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=PRIMARY,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 higher to create an instance fleet cluster, multiple custom AMIs, multiple instance types, On-Demand primary, On-Demand core, multiple core and task nodes

```
aws emr create-cluster --instance-fleets
InstanceFleetType=PRIMARY,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=m5.xlarg
 CustomAmiId=ami-123456}']
InstanceFleetType=CORE,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=m5.xlarge,C
{InstanceType=m6g.xlarge, CustomAmiId=ami-345678}']
InstanceFleetType=TASK,TargetSpotCapacity=1,InstanceTypeConfigs=['{InstanceType=m5.xlarge,Custo
{InstanceType=m6g.xlarge, CustomAmiId=ami-567890}']
```

Example - Use the AWS CLI version 1.20.21 or higher to add task nodes to a running cluster with multiple instance types and multiple custom AMIs

```
aws emr create-cluster --instance-fleets
InstanceFleetType=PRIMARY,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=m5.xlarg
 CustomAmiId=ami-123456}']
InstanceFleetType=CORE,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=m5.xlarge,C
{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,Custo
{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.

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

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

With 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*. 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.

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 with 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.

JSON

```
{
  "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 higher, when you create the AMI.

To add the service role for Amazon EC2 to the list of encryption key users with 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 with 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 lower.

> **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. 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.

Best practices and considerations

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

- The Amazon EMR 7.x series are based on Amazon Linux 2023. For these Amazon EMR versions, you need to use images based on Amazon Linux 2023 for custom AMIs. To find a base custom AMI, see Finding a Linux AMI.
- For Amazon EMR versions lower than 7.x, Amazon Linux 2023 AMIs are not supported.
- Amazon EMR 5.30.0 and higher, 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 lower 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 with Amazon EMR.
- 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.
- If you create a custom AMI that includes the Amazon EC2 Systems Manager (SSM) agent, the enabled SSM agent can cause a provisioning error on the cluster. To avoid this, disable the SSM agent when you use a custom AMI. To do this, when you choose and launch your Amazon EC2 instance, disable the SSM agent prior to using the instance to create a custom AMI and subsequently creating your EMR cluster.

For more information, see Creating an Amazon EBS-backed Linux AMI in the *Amazon EC2 User Guide*.

Change the Amazon Linux release when you create an EMR cluster

When you launch a cluster using Amazon EMR 6.6.0 or higher, it automatically uses the latest Amazon Linux 2 release that has been validated for the default Amazon EMR AMI. You can specify a different Amazon Linux release for your cluster with the Amazon EMR console or the AWS CLI.

Amazon EMR console

To change the Amazon Linux release when you create a cluster from the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose **Create cluster**.
3. For **EMR version**, choose **emr-6.6.0** or higher.
4. Under **Operating system options**, choose **Amazon Linux release**, uncheck the **Automatically apply latest Amazon Linux updates** check box, and choose a desired **Amazon Linux release**.
5. Choose any other options that apply to your cluster.
6. To launch your cluster, choose **Create cluster**.

AWS CLI

To change the Amazon Linux release when you create a cluster with the 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
```

Customizing the Amazon EBS root device volume

You can set your volume type and other attributes, depending on your use case and cost requirements. You can accept default values or make customizations.

EBS root volume defaults

With Amazon EMR 4.x and higher, you can specify the root volume size when you create a cluster. With Amazon EMR releases 6.15.0 and higher, you can also specify the root volume IOPS and throughput. The attributes apply only to the Amazon EBS root device volume, and apply to all instances in the cluster. The attributes don't apply to storage volumes, which you specify separately for each instance type when you create your cluster.

- The default root volume size is 15 GiB in Amazon EMR 6.10.0 and higher. Earlier releases have a default root volume size of 10 GiB. You can adjust this up to 100 GiB.
- The default root volume IOPS is 3000. You can adjust this up to 16000.
- The default root volume throughput is 125 MiB/s. You can adjust this up to 1000 Mib/s.

> **Note**
>
> The root volume size and IOPS can't have a ratio higher than 1 volume to 500 IOPS (1:500), while root volume IOPS and throughput can't have a ratio higher than 1 IOPS to 0.25 throughput (1:0.25).

For more information about Amazon EBS, see Amazon EC2 root device volume.

Root device volume type with the default AMI

When you use the default AMI, the root device volume type is determined by the Amazon EMR release that you use.

- With Amazon EMR releases 6.15.0 and higher, Amazon EMR attaches **General Purpose SSD (gp3)** as the root device volume type.
- With Amazon EMR releases lower than 6.15.0, Amazon EMR attaches **General Purpose SSD (gp2)** as the root device volume type.

Root device volume type with the custom AMI

A custom AMI might have different root device volume types. Amazon EMR always uses your custom AMI volume type.

- With Amazon EMR releases 6.15.0 and higher, you can configure root volume size, IOPS, and throughput for your custom AMI, provided tht these attributes are applicable to the custom AMI volume type.
- With Amazon EMR releases lower than 6.15.0, you can only configure the root volume size for your custom AMI.

If you do not configure root volume size, IOPS, or throughput when you create your cluster, Amazon EMR uses the values from the custom AMI if applicable. If you decide to configure these values when you create your cluster, Amazon EMR uses the values that you specify as long as the values are compatible with and supported by the custom AMI root volume. For more information, see Using a custom AMI to provide more flexibility for Amazon EMR cluster configuration.

Root device volume size pricing

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 might want to consider this in your estimates (1 GB is 0.931323 GiB).

General Purpose SSD gp2 and gp3 are billed differently. To estimate the charges associated with EBS root device volumes in your cluster, use the following formulas:

General Purpose SSD gp2

Cost for **gp2** includes only the EBS volume size in GB.

```
($EBS size in GB/month) * 0.931323 / 30 / 24 * EMR_EBSRootVolumesizeInGiB *
 InstanceCount
```

For example, take a cluster that has a primary 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).

General Purpose SSD gp3

Cost for **gp3** includes EBS volume size in GB, IOPS above 3000 (3000 IOPS free), and throughput above 125 MB/s (125 MB/s free).

```
($EBS size in GB/month) * 0.931323 / 30 / 24 * EMR_EBSRootVolumesizeInGiB *
 InstanceCount
+
($EBS IOPS/Month)/30/24* (EMR_EBSRootVolumeIops - 3000) * InstanceCount
+
($EBS throughput/Month)/30/24* (EMR_EBSRootVolumeThroughputInMb/s - 125) *
 InstanceCount
```

For example, take a cluster that has a primary node, a core node, and uses the base Amazon Linux AMI, with the default 15 GiB root device volume Size, 4000 IOPS, and 140 throughput. If the EBS cost in the Region is USD $0.10/GB/month, $0.005/provisioned IOPS/month over 3000, and $0.040/provisioned MB/s/month over 125. That works out to be approximately $0.009293 per instance per hour, and $0.018586 per hour for the cluster.

Specifying custom root device volume settings

> **Note**
>
> The root volume size and IOPS can't have a ratio higher than 1 volume to 500 IOPS (1:500), while root volume IOPS and throughput can't have a ratio higher than 1 IOPS to 0.25 throughput (1:0.25).

Console

To specify Amazon EBS root device volume attributes from the Amazon EMR console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose **Create cluster**.
3. Select Amazon EMR release 6.15.0 or higher.
4. Under **Cluster configuration**, navigate to the **EBS root volume** section and enter a value for any of the attributes that you want to configure.
5. Choose any other options that apply to your cluster.
6. To launch your cluster, choose **Create cluster**.

CLI

To specify Amazon EBS root device volume attributes with the AWS CLI

- Use the `--ebs-root-volume-size`, `--ebs-root-volume-iops`, and `--ebs-root-volume-throughput` parameters 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-6.15.0\
--ebs-root-volume-size 20 \
--ebs-root-volume-iops 3000\
--ebs-root-volume-throughput 135\
--instance-groups InstanceGroupType=MASTER,\
InstanceCount=1,InstanceType=m5.xlarge
 InstanceGroupType=CORE,InstanceCount=2,InstanceType=m5.xlarge
```

Configure applications when you launch your Amazon EMR cluster

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 with an Amazon EMR cluster

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.

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.

```
--bootstrap-actions Path="s3://amzn-s3-demo-bucket/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 Amazon EMR cluster primary node using SSH.

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
- Use a custom bootstrap action to copy an object from Amazon S3 to each node

Add custom bootstrap actions

Console

To create a cluster with a bootstrap action with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose **Create cluster**.
3. Under **Bootstrap actions**, choose **Add** to specify a name, script location, and optional arguments for your action. Select **Add bootstrap action**.
4. Optionally, add more bootstrap actions.
5. Choose any other options that apply to your cluster.
6. To launch your cluster, choose **Create cluster**.

CLI

To create a cluster with a custom bootstrap action with 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 doesn't 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 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 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.

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://amzn-s3-demo-bucket/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-7.10.0 \
--use-default-roles --ec2-attributes KeyName=myKey \
--applications Name=Hive Name=Pig \
--instance-count 3 --instance-type m5.xlarge \
--bootstrap-actions Path="s3://amzn-s3-demo-bucket/myscriptfolder/copymyfile.sh"
```

Configure Amazon EMR 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.

- **Node types** – Amazon EC2 instances in an EMR cluster are organized into *node types*. There are three: *primary 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 primary node. With a single-node cluster, all components run on the primary node. For more information, see Understand node types in Amazon EMR: primary, core, and task nodes.
- **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 instance types for use with Amazon EMR.

- **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 in a VPC for Amazon EMR.
- **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 an Amazon EMR cluster with instance fleets or uniform instance groups.

> **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 in Amazon EMR: primary, 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.

Primary node

The primary node manages the cluster and typically runs primary components of distributed applications. For example, the primary 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 primary node over SSH as the Hadoop user. For more information, see Connect to the Amazon EMR cluster primary node using SSH. Connecting to the primary node allows you to access directories and files, such as Hadoop log files, directly. For more information, see View Amazon EMR log files. You can also view user interfaces that applications publish as websites running on the primary node. For more information, see View web interfaces hosted on Amazon EMR clusters.

> **Note**
>
> With Amazon EMR 5.23.0 and later, you can launch a cluster with three primary nodes to support high availability of applications like YARN Resource Manager, HDFS NameNode, Spark, Hive, and Ganglia. The primary node is no longer a potential single point of failure with this feature. If one of the primary nodes fails, Amazon EMR automatically fails over to a standby primary node and replaces the failed primary node with a new one with the same configuration and bootstrap actions. For more information, see Plan and Configure Primary Nodes.

Core nodes

Core nodes are managed by the primary 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 Use Amazon EMR cluster scaling to adjust for changing workloads.

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.

> **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?.

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 primary 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'`

Starting with the Amazon EMR 7.x release series, Amazon EMR assigns YARN node labels to instances by their market type, such as On-Demand or Spot. You can enable node labels and restrict application processes to ON_DEMAND by configuring the following properties:

```
yarn.node-labels.enabled: true
yarn.node-labels.am.default-node-label-expression: 'ON_DEMAND'
```

If you're using Amazon EMR 7.0 or higher, you can restrict application process to nodes with the CORE label using the following configuration:

```
yarn.node-labels.enabled: true
yarn.node-labels.am.default-node-label-expression: 'CORE'
```

For Amazon EMR releases 7.2 and higher, if your cluster uses managed scaling with node labels, Amazon EMR will try to scale the cluster based on application process and executor demand independently.

For example, if you use Amazon EMR releases 7.2 or higher and restrict application process to ON_DEMAND nodes, managed scaling scales up ON_DEMAND nodes if application process demand increases. Similarly, if you restrict application process to CORE nodes, managed scaling scales up CORE nodes if application process demand increases.

For information about specific properties, see Amazon EMR settings to prevent job failure because of task node Spot Instance termination.

Configure Amazon EC2 instance types for use with Amazon EMR

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 an Amazon EMR cluster with instance fleets or uniform instance groups. For guidance about choosing instance types and purchasing options for your application, see Configuring Amazon EMR cluster instance types and best practices for Spot instances.

> **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 with Amazon EMR
- Configure networking in a VPC for Amazon EMR
- Create an Amazon EMR cluster with instance fleets or uniform instance groups

Supported instance types with Amazon EMR

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 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.

- Because of a lack of hardware support and default settings that can lead to underutilization of memory and cores, we don't recommend that you use the instance types `c7a`, `c7i`, `m7i`, `m7i-flex`, `r7a`, `r7i`, `r7iz`, `i4i.12xlarge`, `i4i.24xlarge` if you run Amazon EMR releases lower than 5.36.1 and 6.10.0. If you run these instance types in those releases, you might experience lower performance, and you won't see the expected benefits of newer instance types, such as `c7i` vs `c6i`. For optimal resource utilization and performance with these performance types, you should run 5.36.1 and higher or 6.10.0 and higher to maximize their capabilities.
- 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 list the Amazon EC2 instance types that Amazon EMR supports, organized by AWS Region. The tables also list the earliest Amazon EMR releases in the 5.x, 6.x, and 7.x series that support each instance type.

US East (N. Virginia) - us-east-1

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.12xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.16xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8gd.xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.2xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.4xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.8xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.12xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m8gd.16xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.24xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.48xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
Compute Optimized	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5ad.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7gn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.12xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.16xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8gd.xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.2xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.4xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.8xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.12xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.16xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.24xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.48xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
Accelerated Computing	f2.6xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	f2.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	f2.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g6.xlarge	emr-7.2.0
	g6.2xlarge	emr-7.2.0
	g6.4xlarge	emr-7.2.0
	g6.8xlarge	emr-7.2.0
	g6.12xlarge	emr-7.2.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g6.16xlarge	emr-7.2.0
	g6.24xlarge	emr-7.2.0
	g6.48xlarge	emr-7.2.0
	g6e.xlarge	emr-7.5.0
	g6e.2xlarge	emr-7.5.0
	g6e.4xlarge	emr-7.5.0
	g6e.8xlarge	emr-7.5.0
	g6e.12xlarge	emr-7.5.0
	g6e.16xlarge	emr-7.5.0
	g6e.24xlarge	emr-7.5.0
	g6e.48xlarge	emr-7.5.0
	gr6.4xlarge	emr-7.2.0
	gr6.8xlarge	emr-7.2.0
	p2.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p4d.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	p5.48xlarge	emr-6.14.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8gd.xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.2xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.4xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.8xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.12xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.16xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.24xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r8gd.48xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
Storage Optimized	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	h1.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	h1.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	h1.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	h1.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i7i.xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	i7i.2xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	i7i.4xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	i7i.8xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	i7i.12xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	i7i.16xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	i7i.24xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	i7i.48xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i7ie.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.3xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.6xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.18xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

US East (Ohio) - us-east-2

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7a.2xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7a.4xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7a.8xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7a.12xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7a.16xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7a.24xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7a.32xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7a.48xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i-flex.xlarge	emr-4.6.0, emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.6.0, emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.6.0, emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.6.0, emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7i-flex.12xlarge	emr-4.6.0, emr-5.0.1, emr-6.0.0, emr-7.0.0
	m7i-flex.16xlarge	emr-4.6.0, emr-5.0.1, emr-6.0.0, emr-7.0.0
	m8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8gd.xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.2xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.4xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.8xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.12xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.16xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.24xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.48xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
Compute Optimized	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5ad.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7a.xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7a.2xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7a.4xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7a.8xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7a.12xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7a.16xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7a.24xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7a.32xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7a.48xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7gn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7i-flex.xlarge	emr-4.6.0, emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7i-flex.2xlarge	emr-4.6.0, emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7i-flex.4xlarge	emr-4.6.0, emr-5.0.1, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7i-flex.8xlarge	emr-4.6.0, emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7i-flex.12xlarge	emr-4.6.0, emr-5.0.1, emr-6.0.0, emr-7.0.0
	c7i-flex.16xlarge	emr-4.6.0, emr-5.0.1, emr-6.0.0, emr-7.0.0
	c8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8gd.xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c8gd.2xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.4xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.8xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.12xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.16xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.24xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.48xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
Accelerated Computing	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g6.xlarge	emr-7.2.0
	g6.2xlarge	emr-7.2.0
	g6.4xlarge	emr-7.2.0
	g6.8xlarge	emr-7.2.0
	g6.12xlarge	emr-7.2.0
	g6.16xlarge	emr-7.2.0
	g6.24xlarge	emr-7.2.0
	g6.48xlarge	emr-7.2.0
	g6e.xlarge	emr-7.5.0
	g6e.2xlarge	emr-7.5.0
	g6e.4xlarge	emr-7.5.0
	g6e.8xlarge	emr-7.5.0
	g6e.12xlarge	emr-7.5.0
	g6e.16xlarge	emr-7.5.0
	g6e.24xlarge	emr-7.5.0
	g6e.48xlarge	emr-7.5.0
	gr6.4xlarge	emr-7.2.0
	gr6.8xlarge	emr-7.2.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	p2.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p4d.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	p5.48xlarge	emr-6.14.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7a.xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7a.2xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7a.4xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7a.8xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7a.12xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7a.16xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7a.24xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7a.32xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7a.48xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7i.4xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7iz.xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7iz.2xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7iz.4xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7iz.8xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7iz.12xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r7iz.16xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7iz.32xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	r8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8gd.xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.2xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.4xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r8gd.8xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.12xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.16xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.24xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.48xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
Storage Optimized	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	h1.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	h1.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	h1.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	h1.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.1, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i7i.xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	i7i.2xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	i7i.4xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	i7i.8xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	i7i.12xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	i7i.16xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i7i.24xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	i7i.48xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	i7ie.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.3xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.6xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.18xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

US West (N. California) - us-west-1

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.12xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i-flex.16xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.12xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.16xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Accelerated Computing	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	p5.48xlarge	emr-6.14.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
Storage Optimized	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

US West (Oregon) - us-west-2

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.12xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.16xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8gd.xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.2xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.4xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.8xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m8gd.12xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.16xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.24xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.48xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
Compute Optimized	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5ad.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.12xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.16xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8gd.xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.2xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.4xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.8xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.12xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.16xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.24xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c8gd.48xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
Accelerated Computing	f2.6xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	f2.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	f2.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g6.xlarge	emr-7.2.0
	g6.2xlarge	emr-7.2.0
	g6.4xlarge	emr-7.2.0
	g6.8xlarge	emr-7.2.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g6.12xlarge	emr-7.2.0
	g6.16xlarge	emr-7.2.0
	g6.24xlarge	emr-7.2.0
	g6.48xlarge	emr-7.2.0
	g6e.xlarge	emr-7.5.0
	g6e.2xlarge	emr-7.5.0
	g6e.4xlarge	emr-7.5.0
	g6e.8xlarge	emr-7.5.0
	g6e.12xlarge	emr-7.5.0
	g6e.16xlarge	emr-7.5.0
	g6e.24xlarge	emr-7.5.0
	g6e.48xlarge	emr-7.5.0
	gr6.4xlarge	emr-7.2.0
	gr6.8xlarge	emr-7.2.0
	p2.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p4d.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	p5.48xlarge	emr-6.14.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8gd.xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.2xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.4xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.8xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.12xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.16xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r8gd.24xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.48xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
Storage Optimized	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	d3en.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	h1.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	h1.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	h1.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	h1.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i7i.xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	i7i.2xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	i7i.4xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	i7i.8xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	i7i.12xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	i7i.16xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	i7i.24xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i7i.48xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	i7ie.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.3xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.6xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.18xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

AWS GovCloud (US-West) - us-gov-west-1

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.12xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.16xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.12xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.16xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
Accelerated Computing	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g6.xlarge	emr-7.2.0
	g6.2xlarge	emr-7.2.0
	g6.4xlarge	emr-7.2.0
	g6.8xlarge	emr-7.2.0
	g6.12xlarge	emr-7.2.0
	g6.16xlarge	emr-7.2.0
	g6.24xlarge	emr-7.2.0
	g6.48xlarge	emr-7.2.0
	gr6.4xlarge	emr-7.2.0
	gr6.8xlarge	emr-7.2.0
	p2.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p4d.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Storage Optimized	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

AWS GovCloud (US-East) - us-gov-east-1

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5d.16xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i.24xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m7i-flex.12xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	m7i-flex.16xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5.12xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5d.xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5d.4xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7i.24xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
Accelerated Computing	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5.12xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5d.4xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Storage Optimized	i3.xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.16.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Africa (Cape Town) - af-south-1

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5d.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5ad.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5d.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5d.18xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Accelerated Computing	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5d.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5d.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Storage Optimized	i3.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Asia Pacific (Hong Kong) - ap-east-1

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5d.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5d.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
Accelerated Computing	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Storage Optimized	i3.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Asia Pacific (Jakarta) - ap-southeast-3

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m5.2xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m5.4xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m5.8xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m5.12xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m5.16xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m5.24xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5d.xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m5d.2xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m5d.4xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m5d.8xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m5d.12xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m5d.16xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m5d.24xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m6g.xlarge	emr-5.30.2, emr-6.1.1, emr-7.0.0
	m6g.2xlarge	emr-5.30.2, emr-6.1.1, emr-7.0.0
	m6g.4xlarge	emr-5.30.2, emr-6.1.1, emr-7.0.0
	m6g.8xlarge	emr-5.30.2, emr-6.1.1, emr-7.0.0
	m6g.12xlarge	emr-5.30.2, emr-6.1.1, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6g.16xlarge	emr-5.30.2, emr-6.1.1, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m7i.2xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m7i.4xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i.8xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m7i.12xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m7i.16xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m7i.24xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m7i.48xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m7i-flex.xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m7i-flex.2xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m7i-flex.4xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	m7i-flex.8xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5.2xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5.4xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5.9xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5.12xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5.18xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5.24xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5d.xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5d.2xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5d.4xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5d.9xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5d.12xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5d.18xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5d.24xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5n.xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5n.2xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5n.4xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5n.9xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c5n.18xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	c6g.xlarge	emr-5.31.1, emr-6.1.1, emr-7.0.0
	c6g.2xlarge	emr-5.31.1, emr-6.1.1, emr-7.0.0
	c6g.4xlarge	emr-5.31.1, emr-6.1.1, emr-7.0.0
	c6g.8xlarge	emr-5.31.1, emr-6.1.1, emr-7.0.0
	c6g.12xlarge	emr-5.31.1, emr-6.1.1, emr-7.0.0
	c6g.16xlarge	emr-5.31.1, emr-6.1.1, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Accelerated Computing	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r5.2xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r5.4xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r5.8xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r5.12xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r5.16xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r5.24xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r5d.xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r5d.2xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r5d.4xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5d.8xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r5d.12xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r5d.16xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r5d.24xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r6g.xlarge	emr-5.31.1, emr-6.1.1, emr-7.0.0
	r6g.2xlarge	emr-5.31.1, emr-6.1.1, emr-7.0.0
	r6g.4xlarge	emr-5.31.1, emr-6.1.1, emr-7.0.0
	r6g.8xlarge	emr-5.31.1, emr-6.1.1, emr-7.0.0
	r6g.12xlarge	emr-5.31.1, emr-6.1.1, emr-7.0.0
	r6g.16xlarge	emr-5.31.1, emr-6.1.1, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r7i.2xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r7i.4xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r7i.8xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r7i.12xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r7i.16xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r7i.24xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	r7i.48xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Storage Optimized	d3en.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	d3en.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	i3.xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i3.2xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i3.4xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i3.8xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i3.16xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i3en.xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i3en.2xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i3en.3xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3en.6xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i3en.12xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i3en.24xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.30.2, emr-6.0.1, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Asia Pacific (Melbourne) - ap-southeast-4

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m5.2xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m5.4xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m5.8xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m5.12xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m5.16xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m5.24xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m5d.xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m5d.2xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m5d.4xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m5d.8xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m5d.12xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5d.16xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m5d.24xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m6g.2xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m6g.4xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m6g.8xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m6g.12xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m6g.16xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m6gd.xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m6gd.2xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m6gd.4xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m6gd.8xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6gd.12xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m6gd.16xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m7i.2xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m7i.4xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m7i.8xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i.12xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m7i.16xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m7i.24xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m7i.48xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m7i-flex.xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m7i-flex.2xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m7i-flex.4xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m7i-flex.8xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m7i-flex.12xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	m7i-flex.16xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	c5.2xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5.4xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	c5.9xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	c5.12xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	c5.18xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	c5.24xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	c5d.xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	c5d.2xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	c5d.4xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	c5d.9xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	c5d.12xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	c5d.18xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	c5d.24xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6g.xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	c6g.2xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	c6g.4xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	c6g.8xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	c6g.12xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	c6g.16xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r5.2xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r5.4xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r5.8xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r5.12xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r5.16xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r5.24xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r5d.xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r5d.2xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r5d.4xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5d.8xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r5d.12xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r5d.16xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r5d.24xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r6g.xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r6g.2xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r6g.4xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r6g.8xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r6g.12xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r6g.16xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r7i.xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r7i.2xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7i.4xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r7i.8xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r7i.12xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r7i.16xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r7i.24xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	r7i.48xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Storage Optimized	i3.xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	i3.2xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	i3.4xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3.8xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	i3.16xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	i3en.xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	i3en.2xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	i3en.3xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	i3en.6xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	i3en.12xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	i3en.24xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.12xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.36.0, emr-6.8.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Asia Pacific (Malaysia) - ap-southeast-5

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m6g.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6g.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6g.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6g.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6g.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6g.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6gd.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6gd.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6gd.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6gd.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6gd.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6gd.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6i.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6i.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6i.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6i.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6i.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6i.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6i.24xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6i.32xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6id.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6id.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6id.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6id.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6id.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6id.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6id.24xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m6id.32xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m7g.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m7g.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7g.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m7g.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m7g.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m7g.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m7gd.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m7gd.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m7gd.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m7gd.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m7gd.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m7gd.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m7i.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m7i.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m7i.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m7i.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m7i.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m7i.24xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m7i.48xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m7i-flex.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m7i-flex.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m7i-flex.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m7i-flex.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m7i-flex.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	m7i-flex.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
Compute Optimized	c6g.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6g.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6g.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6g.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6g.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6g.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6gn.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6gn.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6gn.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6gn.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6gn.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6gn.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6i.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6i.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6i.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6i.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6i.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6i.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6i.24xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6i.32xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6id.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6id.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6id.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6id.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6id.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6id.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6id.24xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6id.32xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6in.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6in.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6in.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6in.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6in.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6in.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6in.24xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c6in.32xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7g.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c7g.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c7g.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c7g.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c7g.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c7g.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c7gd.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c7gd.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c7gd.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c7gd.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c7gd.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c7gd.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7i.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c7i.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c7i.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c7i.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c7i.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c7i.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c7i.24xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c7i.48xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c7i-flex.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c7i-flex.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c7i-flex.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c7i-flex.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7i-flex.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	c7i-flex.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
Accelerated Computing	g6.xlarge	emr-7.2.0
	g6.2xlarge	emr-7.2.0
	g6.4xlarge	emr-7.2.0
	g6.8xlarge	emr-7.2.0
	g6.12xlarge	emr-7.2.0
	g6.16xlarge	emr-7.2.0
	g6.24xlarge	emr-7.2.0
	g6.48xlarge	emr-7.2.0
	gr6.4xlarge	emr-7.2.0
	gr6.8xlarge	emr-7.2.0
Memory Optimized	r6g.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r6g.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r6g.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r6g.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6g.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r6g.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r6i.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r6i.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r6i.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r6i.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r6i.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r6i.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r6i.24xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r6i.32xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r6id.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r6id.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6id.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r6id.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r6id.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r6id.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r6id.24xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r6id.32xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r7g.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r7g.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r7g.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r7g.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r7g.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r7g.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7gd.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r7gd.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r7gd.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r7gd.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r7gd.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r7gd.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r7i.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r7i.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r7i.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r7i.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r7i.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r7i.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7i.24xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	r7i.48xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	x2idn.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	x2idn.24xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	x2idn.32xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	x2iedn.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
Storage Optimized	i3en.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	i3en.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	i3en.3xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	i3en.6xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	i3en.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	i3en.24xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	i4i.xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	i4i.2xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	i4i.4xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	i4i.8xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	i4i.12xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	i4i.16xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.24xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0
	i4i.32xlarge	emr-5.36.2, emr-6.11.1, emr-7.0.0

Asia Pacific (Mumbai) - ap-south-1

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.6.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.6.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.6.0, emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i-flex.8xlarge	emr-4.6.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.12xlarge	emr-4.6.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.16xlarge	emr-4.6.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.xlarge	emr-4.6.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.2xlarge	emr-4.6.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.4xlarge	emr-4.6.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.8xlarge	emr-4.6.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.12xlarge	emr-4.6.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.16xlarge	emr-4.6.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Accelerated Computing	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g6.xlarge	emr-7.2.0
	g6.2xlarge	emr-7.2.0
	g6.4xlarge	emr-7.2.0
	g6.8xlarge	emr-7.2.0
	g6.12xlarge	emr-7.2.0
	g6.16xlarge	emr-7.2.0
	g6.24xlarge	emr-7.2.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g6.48xlarge	emr-7.2.0
	gr6.4xlarge	emr-7.2.0
	gr6.8xlarge	emr-7.2.0
	p2.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p4d.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	p5.48xlarge	emr-6.14.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
Storage Optimized	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Asia Pacific (Hyderabad) - ap-south-2

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5d.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6i.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.32xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5.9xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.18xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.9xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.18xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6i.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6i.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6i.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6i.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6i.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6i.32xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
Memory Optimized	r5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5d.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6i.32xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Storage Optimized	i3.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.3xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.6xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Asia Pacific (Osaka) - ap-northeast-3

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
Compute Optimized	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
Accelerated Computing	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Storage Optimized	i3.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Asia Pacific (Seoul) - ap-northeast-2

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.12xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.16xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.12xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7i-flex.16xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
Accelerated Computing	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g6.xlarge	emr-7.2.0
	g6.2xlarge	emr-7.2.0
	g6.4xlarge	emr-7.2.0
	g6.8xlarge	emr-7.2.0
	g6.12xlarge	emr-7.2.0
	g6.16xlarge	emr-7.2.0
	g6.24xlarge	emr-7.2.0
	g6.48xlarge	emr-7.2.0
	g6e.xlarge	emr-7.5.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g6e.2xlarge	emr-7.5.0
	g6e.4xlarge	emr-7.5.0
	g6e.8xlarge	emr-7.5.0
	g6e.12xlarge	emr-7.5.0
	g6e.16xlarge	emr-7.5.0
	g6e.24xlarge	emr-7.5.0
	g6e.48xlarge	emr-7.5.0
	gr6.4xlarge	emr-7.2.0
	gr6.8xlarge	emr-7.2.0
	p2.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p4d.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
Memory Optimized	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
Storage Optimized	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Asia Pacific (Singapore) - ap-southeast-1

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.12xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.16xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5ad.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.12xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.16xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
Accelerated Computing	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	p2.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
Storage Optimized	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	d3en.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Asia Pacific (Sydney) - ap-southeast-2

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.12xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i-flex.16xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5ad.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.12xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.16xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Accelerated Computing	f2.6xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	f2.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	f2.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g6.xlarge	emr-7.2.0
	g6.2xlarge	emr-7.2.0
	g6.4xlarge	emr-7.2.0
	g6.8xlarge	emr-7.2.0
	g6.12xlarge	emr-7.2.0
	g6.16xlarge	emr-7.2.0
	g6.24xlarge	emr-7.2.0
	g6.48xlarge	emr-7.2.0
	gr6.4xlarge	emr-7.2.0
	gr6.8xlarge	emr-7.2.0
	p2.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	p2.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p4d.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	p5.48xlarge	emr-6.14.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
Storage Optimized	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	im4gn.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Asia Pacific (Tokyo) - ap-northeast-1

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i-flex.12xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.16xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7gn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.12xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.16xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Accelerated Computing	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g6.xlarge	emr-7.2.0
	g6.2xlarge	emr-7.2.0
	g6.4xlarge	emr-7.2.0
	g6.8xlarge	emr-7.2.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g6.12xlarge	emr-7.2.0
	g6.16xlarge	emr-7.2.0
	g6.24xlarge	emr-7.2.0
	g6.48xlarge	emr-7.2.0
	g6e.xlarge	emr-7.5.0
	g6e.2xlarge	emr-7.5.0
	g6e.4xlarge	emr-7.5.0
	g6e.8xlarge	emr-7.5.0
	g6e.12xlarge	emr-7.5.0
	g6e.16xlarge	emr-7.5.0
	g6e.24xlarge	emr-7.5.0
	g6e.48xlarge	emr-7.5.0
	gr6.4xlarge	emr-7.2.0
	gr6.8xlarge	emr-7.2.0
	p2.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p4d.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	p5.48xlarge	emr-6.14.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
Storage Optimized	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	d3en.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i7ie.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i7ie.3xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.6xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.18xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Canada (Central) - ca-central-1

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i.12xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.8.2, emr-5.0.2, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.8.2, emr-5.0.2, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.8.2, emr-5.0.2, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.8.2, emr-5.0.2, emr-6.0.0, emr-7.0.0
	m7i-flex.12xlarge	emr-4.8.2, emr-5.0.2, emr-6.0.0, emr-7.0.0
	m7i-flex.16xlarge	emr-4.8.2, emr-5.0.2, emr-6.0.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	c7i-flex.xlarge	emr-4.8.2, emr-5.0.2, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7i-flex.2xlarge	emr-4.8.2, emr-5.0.2, emr-6.0.0, emr-7.0.0
	c7i-flex.4xlarge	emr-4.8.2, emr-5.0.2, emr-6.0.0, emr-7.0.0
	c7i-flex.8xlarge	emr-4.8.2, emr-5.0.2, emr-6.0.0, emr-7.0.0
	c7i-flex.12xlarge	emr-4.8.2, emr-5.0.2, emr-6.0.0, emr-7.0.0
	c7i-flex.16xlarge	emr-4.8.2, emr-5.0.2, emr-6.0.0, emr-7.0.0
Accelerated Computing	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g6.xlarge	emr-7.2.0
	g6.2xlarge	emr-7.2.0
	g6.4xlarge	emr-7.2.0
	g6.8xlarge	emr-7.2.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g6.12xlarge	emr-7.2.0
	g6.16xlarge	emr-7.2.0
	g6.24xlarge	emr-7.2.0
	g6.48xlarge	emr-7.2.0
	gr6.4xlarge	emr-7.2.0
	gr6.8xlarge	emr-7.2.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p5.48xlarge	emr-6.14.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7i.xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Storage Optimized	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.2, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Canada West (Calgary) - ca-west-1

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5d.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m5d.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m5d.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m5d.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m5d.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m5d.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m5d.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6g.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6g.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6g.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6g.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6g.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6g.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6gd.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6gd.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6gd.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6gd.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6gd.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6gd.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6i.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6i.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6i.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6i.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6i.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6i.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6i.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6i.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c5.9xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c5.18xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6g.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6g.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6g.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6g.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6g.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6g.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6gn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6gn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6gn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6gn.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6gn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6i.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6i.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6i.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6i.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6i.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6i.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6i.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6i.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6id.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c6id.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7g.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6g.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6g.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6g.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6g.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6g.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6g.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6i.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6i.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6i.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6i.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6i.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6i.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6i.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6i.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6id.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6id.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6id.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6id.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Storage Optimized	i3en.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	i3en.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	i3en.3xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	i3en.6xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	i3en.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	i3en.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

China (Ningxia) - cn-northwest-1

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
Accelerated Computing	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
Storage Optimized	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

China (Beijing) - cn-north-1

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
Accelerated Computing	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	p2.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Storage Optimized	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Europe (Frankfurt) - eu-central-1

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.12xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.16xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8gd.xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.2xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.4xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.8xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.12xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.16xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.24xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	m8gd.48xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
Compute Optimized	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5ad.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.12xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.16xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8gd.xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.2xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.4xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.8xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.12xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.16xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	c8gd.24xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c8gd.48xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
Accelerated Computing	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g6.xlarge	emr-7.2.0
	g6.2xlarge	emr-7.2.0
	g6.4xlarge	emr-7.2.0
	g6.8xlarge	emr-7.2.0
	g6.12xlarge	emr-7.2.0
	g6.16xlarge	emr-7.2.0
	g6.24xlarge	emr-7.2.0
	g6.48xlarge	emr-7.2.0
	g6e.xlarge	emr-7.5.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g6e.2xlarge	emr-7.5.0
	g6e.4xlarge	emr-7.5.0
	g6e.8xlarge	emr-7.5.0
	g6e.12xlarge	emr-7.5.0
	g6e.16xlarge	emr-7.5.0
	g6e.24xlarge	emr-7.5.0
	g6e.48xlarge	emr-7.5.0
	gr6.4xlarge	emr-7.2.0
	gr6.8xlarge	emr-7.2.0
	p2.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p4d.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
Memory Optimized	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7iz.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8gd.xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.2xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.4xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.8xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.12xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.16xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.24xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	r8gd.48xlarge	emr-5.36.2, emr-6.15.0, emr-7.1.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
Storage Optimized	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	d3en.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i7ie.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i7ie.3xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.6xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.18xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Europe (Zurich) - eu-central-2

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5d.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6gd.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.32xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.9xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.18xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.9xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.18xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gd.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6gd.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6gd.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6gd.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6gd.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6gd.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
Accelerated Computing	g6.xlarge	emr-7.2.0
	g6.2xlarge	emr-7.2.0
	g6.4xlarge	emr-7.2.0
	g6.8xlarge	emr-7.2.0
	g6.12xlarge	emr-7.2.0
	g6.16xlarge	emr-7.2.0
	g6.24xlarge	emr-7.2.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g6.48xlarge	emr-7.2.0
	gr6.4xlarge	emr-7.2.0
	gr6.8xlarge	emr-7.2.0
Memory Optimized	r5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5d.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6gd.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6gd.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6gd.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6gd.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6gd.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6gd.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.32xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Storage Optimized	d3.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	d3.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	d3.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	d3.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.3xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.6xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Europe (Ireland) - eu-west-1

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.12xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i-flex.16xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5ad.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7i-flex.12xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.16xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Accelerated Computing	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	p2.xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p2.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p4d.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7a.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7a.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7a.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7iz.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7iz.32xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
Storage Optimized	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	d3en.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3en.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	h1.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	h1.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	h1.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	h1.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i7ie.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.3xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.6xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i7ie.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.18xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Europe (London) - eu-west-2

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.8.2, emr-5.0.3, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.8.2, emr-5.0.3, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i-flex.4xlarge	emr-4.8.2, emr-5.0.3, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.8.2, emr-5.0.3, emr-6.0.0, emr-7.0.0
	m8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	c7i-flex.xlarge	emr-4.8.2, emr-5.0.3, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7i-flex.2xlarge	emr-4.8.2, emr-5.0.3, emr-6.0.0, emr-7.0.0
	c7i-flex.4xlarge	emr-4.8.2, emr-5.0.3, emr-6.0.0, emr-7.0.0
	c7i-flex.8xlarge	emr-4.8.2, emr-5.0.3, emr-6.0.0, emr-7.0.0
	c7i-flex.12xlarge	emr-4.8.2, emr-5.0.3, emr-6.0.0, emr-7.0.0
	c7i-flex.16xlarge	emr-4.8.2, emr-5.0.3, emr-6.0.0, emr-7.0.0
	c8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Accelerated Computing	f2.6xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	f2.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	f2.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	g3.4xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.8xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3.16xlarge	emr-5.18.0, emr-6.0.0, emr-7.0.0
	g3s.xlarge	emr-5.19.0, emr-6.0.0, emr-7.0.0
	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g6.xlarge	emr-7.2.0
	g6.2xlarge	emr-7.2.0
	g6.4xlarge	emr-7.2.0
	g6.8xlarge	emr-7.2.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g6.12xlarge	emr-7.2.0
	g6.16xlarge	emr-7.2.0
	g6.24xlarge	emr-7.2.0
	g6.48xlarge	emr-7.2.0
	gr6.4xlarge	emr-7.2.0
	gr6.8xlarge	emr-7.2.0
	p3.2xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.8xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p3.16xlarge	emr-5.10.0, emr-6.0.0, emr-7.0.0
	p5.48xlarge	emr-6.14.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7i.48xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	z1d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.3xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.6xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	z1d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
Storage Optimized	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.3, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i7ie.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.3xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.6xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.18xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i7ie.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i7ie.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Europe (Milan) - eu-south-1

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5a.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5ad.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5d.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5d.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
Accelerated Computing	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5a.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5d.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7i.48xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Storage Optimized	i3.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.29.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Europe (Spain) - eu-south-2

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7a.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7a.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7a.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7a.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7a.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7a.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7a.32xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7a.48xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7i.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7i.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7i.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7i.48xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7i-flex.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7i-flex.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7i-flex.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7i-flex.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7i-flex.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i-flex.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5.9xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.18xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.9xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.18xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6gd.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6gd.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6gd.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6gd.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6gd.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6gd.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7a.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7a.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7a.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7a.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7a.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7a.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7a.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7a.32xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7a.48xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7i.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7i.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7i.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7i.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7i.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7i.48xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7i-flex.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7i-flex.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7i-flex.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7i-flex.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7i-flex.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c7i-flex.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Accelerated Computing	g6.xlarge	emr-7.2.0
	g6.2xlarge	emr-7.2.0
	g6.4xlarge	emr-7.2.0
	g6.8xlarge	emr-7.2.0
	g6.12xlarge	emr-7.2.0
	g6.16xlarge	emr-7.2.0
	g6.24xlarge	emr-7.2.0
	g6.48xlarge	emr-7.2.0
	g6e.xlarge	emr-7.5.0
	g6e.2xlarge	emr-7.5.0
	g6e.4xlarge	emr-7.5.0
	g6e.8xlarge	emr-7.5.0
	g6e.12xlarge	emr-7.5.0
	g6e.16xlarge	emr-7.5.0
	g6e.24xlarge	emr-7.5.0
	g6e.48xlarge	emr-7.5.0
	gr6.4xlarge	emr-7.2.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	gr6.8xlarge	emr-7.2.0
Memory Optimized	r5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5d.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6gd.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6gd.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6gd.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6gd.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6gd.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6gd.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r7a.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7a.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7a.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7a.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7a.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7a.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7a.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7a.32xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7a.48xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7i.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7i.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7i.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7i.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7i.48xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Storage Optimized	i3.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.3xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.6xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	im4gn.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	im4gn.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	im4gn.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	im4gn.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	im4gn.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Europe (Paris) - eu-west-3

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i.xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.9.2, emr-5.5.3, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.9.2, emr-5.5.3, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.9.2, emr-5.5.3, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.9.2, emr-5.5.3, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i-flex.12xlarge	emr-4.9.2, emr-5.5.3, emr-6.0.0, emr-7.0.0
	m7i-flex.16xlarge	emr-4.9.2, emr-5.5.3, emr-6.0.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7i.xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	c7i-flex.xlarge	emr-4.9.2, emr-5.5.3, emr-6.0.0, emr-7.0.0
	c7i-flex.2xlarge	emr-4.9.2, emr-5.5.3, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7i-flex.4xlarge	emr-4.9.2, emr-5.5.3, emr-6.0.0, emr-7.0.0
	c7i-flex.8xlarge	emr-4.9.2, emr-5.5.3, emr-6.0.0, emr-7.0.0
	c7i-flex.12xlarge	emr-4.9.2, emr-5.5.3, emr-6.0.0, emr-7.0.0
	c7i-flex.16xlarge	emr-4.9.2, emr-5.5.3, emr-6.0.0, emr-7.0.0
Accelerated Computing	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g6.xlarge	emr-7.2.0
	g6.2xlarge	emr-7.2.0
	g6.4xlarge	emr-7.2.0
	g6.8xlarge	emr-7.2.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g6.12xlarge	emr-7.2.0
	g6.16xlarge	emr-7.2.0
	g6.24xlarge	emr-7.2.0
	g6.48xlarge	emr-7.2.0
	gr6.4xlarge	emr-7.2.0
	gr6.8xlarge	emr-7.2.0
Memory Optimized	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7i.xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	r7i.16xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Storage Optimized	d3.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	d3.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.5.3, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	im4gn.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	im4gn.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	is4gen.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	is4gen.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Europe (Stockholm) - eu-north-1

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7a.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7a.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7a.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7a.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7a.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7a.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7a.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7a.32xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7a.48xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7i.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i.48xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7i-flex.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m7i-flex.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	m8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c5d.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7a.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7a.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7a.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7a.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7a.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7a.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7a.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7a.32xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7a.48xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7i.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7i-flex.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7i-flex.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7i-flex.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7i-flex.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7i-flex.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c7i-flex.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	c8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Accelerated Computing	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g6.xlarge	emr-7.2.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g6.2xlarge	emr-7.2.0
	g6.4xlarge	emr-7.2.0
	g6.8xlarge	emr-7.2.0
	g6.12xlarge	emr-7.2.0
	g6.16xlarge	emr-7.2.0
	g6.24xlarge	emr-7.2.0
	g6.48xlarge	emr-7.2.0
	g6e.xlarge	emr-7.5.0
	g6e.2xlarge	emr-7.5.0
	g6e.4xlarge	emr-7.5.0
	g6e.8xlarge	emr-7.5.0
	g6e.12xlarge	emr-7.5.0
	g6e.16xlarge	emr-7.5.0
	g6e.24xlarge	emr-7.5.0
	g6e.48xlarge	emr-7.5.0
	gr6.4xlarge	emr-7.2.0
	gr6.8xlarge	emr-7.2.0
	p5.48xlarge	emr-6.14.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r5dn.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5dn.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5dn.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6idn.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6idn.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6idn.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7a.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7a.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7a.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7a.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7a.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7a.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7a.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7a.32xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7a.48xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7i.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	r8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Storage Optimized	i3.xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3.8xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.12xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.17.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Israel (Tel Aviv) - il-central-1

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.32xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.9xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.18xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.9xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.18xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6gn.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6gn.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6gn.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6gn.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6gn.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6gn.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6i.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6i.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6i.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6i.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6i.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6i.32xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Accelerated Computing	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6i.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.32xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Storage Optimized	d3.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	d3.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	d3.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	d3.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3en.3xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.6xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Middle East (Bahrain) - me-south-1

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m5.8xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5d.16xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m5d.24xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5.9xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5ad.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5d.xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Accelerated Computing	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r5.2xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5.8xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r5d.xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Storage Optimized	i3.xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3.4xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	i3.8xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.24.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Middle East (UAE) - me-central-1

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6gd.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6i.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m6i.32xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.9xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.18xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.9xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.18xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Accelerated Computing	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5d.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r5d.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6g.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6g.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r6i.32xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
Storage Optimized	i3.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.4xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.8xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3.16xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.2xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.3xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.6xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i3en.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.36.0, emr-6.7.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

South America (São Paulo) - sa-east-1

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
General Purpose	m5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	m5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	m5zn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.3xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.6xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m5zn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6g.xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.2xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.4xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.8xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.12xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6g.16xlarge	emr-5.30.0, emr-6.1.0, emr-7.0.0
	m6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	m6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	m6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	m7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7gd.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	m7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	m7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.12xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	m7i-flex.16xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
Compute Optimized	c5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5a.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5a.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c5ad.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5ad.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5ad.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.9xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5d.18xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	c5n.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.9xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c5n.18xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	c6a.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6a.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6a.48xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	c6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6gn.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	c6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	c6id.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.12xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6id.24xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c6id.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	c6in.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c6in.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c7g.xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.2xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.4xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7g.8xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.12xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7g.16xlarge	emr-5.36.1, emr-6.7.0, emr-7.0.0
	c7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c7i-flex.2xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.4xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.8xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.12xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c7i-flex.16xlarge	emr-4.2.0, emr-5.0.0, emr-6.0.0, emr-7.0.0
	c8g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	c8g.24xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	c8g.48xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
Accelerated Computing	g4dn.xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.2xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.4xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.8xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.12xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g4dn.16xlarge	emr-5.30.0, emr-6.0.0, emr-7.0.0
	g5.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.12xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	g5.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g5.48xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	g6.xlarge	emr-7.2.0
	g6.2xlarge	emr-7.2.0
	g6.4xlarge	emr-7.2.0
	g6.8xlarge	emr-7.2.0
	g6.12xlarge	emr-7.2.0
	g6.16xlarge	emr-7.2.0
	g6.24xlarge	emr-7.2.0
	g6.48xlarge	emr-7.2.0
	gr6.4xlarge	emr-7.2.0
	gr6.8xlarge	emr-7.2.0
	p4d.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	p5.48xlarge	emr-6.14.0, emr-7.0.0
Memory Optimized	r5.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5a.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.8xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5a.16xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5a.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.2xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.4xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.12xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5ad.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5ad.24xlarge	emr-5.20.0, emr-6.0.0, emr-7.0.0
	r5b.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5b.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5b.24xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r5d.xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.2xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.4xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.8xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.12xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.16xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5d.24xlarge	emr-5.13.0, emr-6.0.0, emr-7.0.0
	r5n.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r5n.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r5n.24xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.2xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.4xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.8xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.12xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6g.16xlarge	emr-5.31.0, emr-6.1.0, emr-7.0.0
	r6gd.xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6gd.2xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.4xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.8xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.12xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6gd.16xlarge	emr-5.33.0, emr-6.3.0, emr-7.0.0
	r6i.xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.2xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.4xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.8xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.12xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.16xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r6i.24xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r6i.32xlarge	emr-5.35.0, emr-6.6.0, emr-7.0.0
	r7g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.12xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	r7i.xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.2xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.4xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.8xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	r7i.16xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	r7i.48xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	x1.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x1e.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x1e.32xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	x2idn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	x2idn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2idn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.2xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.4xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.8xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.16xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.24xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
	x2iedn.32xlarge	emr-5.36.1, emr-6.9.0, emr-7.0.0
Storage Optimized	i3.xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.2xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.4xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i3.8xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3.16xlarge	emr-5.9.0, emr-6.0.0, emr-7.0.0
	i3en.xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.2xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.3xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.6xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.12xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i3en.24xlarge	emr-5.25.0, emr-6.0.0, emr-7.0.0
	i4g.xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.2xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.4xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4g.8xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0

Instance class	Instance type	Minimum supported Amazon EMR version (5.x, 6.x, 7.x)
	i4g.16xlarge	emr-5.36.1, emr-6.10.0, emr-7.0.0
	i4i.xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.2xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.4xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.8xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.12xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.16xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0
	i4i.24xlarge	emr-5.0.0, emr-6.0.0, emr-7.0.0
	i4i.32xlarge	emr-5.36.1, emr-6.8.0, emr-7.0.0

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] | m3.xlarge[1] | m3.2xlarge[1] | m4.large | m4.xlarge | m4.2xlarge | m4.4xlarge | m4.10xlarge | m4.16xlarge
Compute Optimized	c1.medium[1] [2] | c1.xlarge[1] | c3.xlarge[1] | c3.2xlarge[1] | c3.4xlarge[1] | c3.8xlarge[1] | c4.large | c4.xlarge | c4.2xlarge | c4.4xlarge | c4.8xlarge
Memory Optimized	m2.xlarge[1] | m2.2xlarge[1] | m2.4xlarge[1] | r3.xlarge | r3.2xlarge | r3.4xlarge | r3.8xlarge | r4.xlarge | r4.2xlarge | r4.4xlarge | r4.8xlarge | r4.16xlarge
Storage Optimized	d2.xlarge | d2.2xlarge | d2.4xlarge | d2.8xlarge | i2.xlarge | i2.2xlarge | i2.4xlarge | i2.8xlarge

[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 in Amazon EMR

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 an Amazon EMR cluster with instance fleets or uniform instance groups.

On-Demand Instances

With On-Demand Instances, you pay for compute capacity by the second. 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*.

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*. 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?.

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*. 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 in a VPC for Amazon EMR.

> **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*.

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 an Amazon EMR cluster with instance fleets or uniform instance groups.

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 Planning and configuring instance fleets for your Amazon EMR cluster.

Instance storage options and behavior in Amazon EMR

Overview

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 might "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 you shouldn't expect data to persist. Although the data is ephemeral, it is possible that data in HDFS could 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).

Considerations

Keep in mind these additional considerations when you use Amazon EBS with EMR clusters:

- 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 to provide more flexibility for Amazon EMR cluster configuration.
- 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.
- 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.
- Amazon EBS has a fixed limit of 2,500 EBS volumes per instance launch request. This limit also applies to Amazon EMR on EC2 clusters. We recommend that you launch clusters with the total number of EBS volumes within this limit, and then manually scale up the cluster or with Amazon EMR managed scaling as needed. To learn more about the EBS volume limit, see Service quotas.

Default Amazon EBS storage for instances

For EC2 instances that have EBS-only storage, Amazon EMR allocates Amazon EBS gp2 or gp3 storage volumes to instances. When you create a cluster with Amazon EMR releases 5.22.0 and higher, the default amount of Amazon EBS storage increases relative to the size of the instance.

We split any increased storage across multiple volumes. This gives increased IOPS performance and, in turn, increased performance for some standardized workloads. If you want to use a different Amazon EBS instance storage configuration, you can specify this when you create an EMR cluster or add nodes to an existing cluster. You can use Amazon EBS gp2 or gp3 volumes as root volumes, and add gp2 or gp3 volumes as additional volumes. For more information, see Specifying additional EBS storage volumes.

The following table identifies the default number of Amazon EBS gp2 storage volumes, sizes, and total sizes per instance type. For information about gp2 volumes compared to gp3, see Comparing Amazon EBS volume types gp2 and gp3.

Default Amazon EBS gp2 storage volumes and size by instance type for Amazon EMR 5.22.0 and higher

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
*.12xlarge	4	192	768
*.16xlarge	4	256	1024
*.18xlarge	4	288	1152

Instance size	Number of volumes	Volume size (GiB)	Total size (GiB)
*.24xlarge	4	384	1536

Default Amazon EBS root volume for instances

With Amazon EMR releases 6.15 and higher, Amazon EMR automatically attaches an Amazon EBS General Purpose SSD (gp3) as the root device for its AMIs to enhance performance. With earlier releases, Amazon EMR attaches EBS General Purpose SSD (gp2) as the root device.

	6.15 and higher	6.14 and lower
Default root volume type		
Default size		
Default IOPS		
Default throughput		

For information on how to customize the Amazon EBS root device volume, see Specifying additional EBS storage volumes.

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 to the analytic and business needs of your applications. For example, some applications might need to spill to disk while others can safely work in-memory or with 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, launch a new EMR cluster. In addition, we don't recommend that you update the throughput and IOPS on a cluster that is in use or that is being provisioned, because Amazon EMR uses the throughput and IOPS values you specify at cluster launch time for any new instance that it adds during cluster scale-up. For more information, see Comparing Amazon EBS volume types gp2 and gp3 and Selecting IOPS and throughput when migrating to gp3 Amazon EBS volume types.

> **Important**
>
> To use a gp3 volume with your EMR cluster, you must launch a new cluster.

Comparing Amazon EBS volume types gp2 and gp3

Here is a comparison of cost between gp2 and gp3 volumes in the US East (N. Virginia) Region. For the most up to date information, see the Amazon EBS General Purpose Volumes product page and the Amazon EBS Pricing Page.

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.

Volume type	gp3	gp2
Max throughput/volume	1,000 MiB/s	250 MiB/s
Price	$0.08/GiB-month 3,000 IOPS free and $0.005/provisioned IOPS-month over 3,000; 125 MiB/s free and $0.04/pro visioned MiB/s-month over 125MiB/s	$0.10/GiB-month

Selecting IOPS and throughput when migrating to gp3 Amazon EBS volume types

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

Volume size	IOPS	Throughput
		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 in a VPC for Amazon EMR

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 when you launch a cluster
- Set up a VPC to host Amazon EMR clusters
- Launch clusters into a VPC with Amazon EMR
- Sample policies for private subnets that access Amazon S3
- More resources for learning about VPCs

Amazon VPC options when you launch a cluster

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

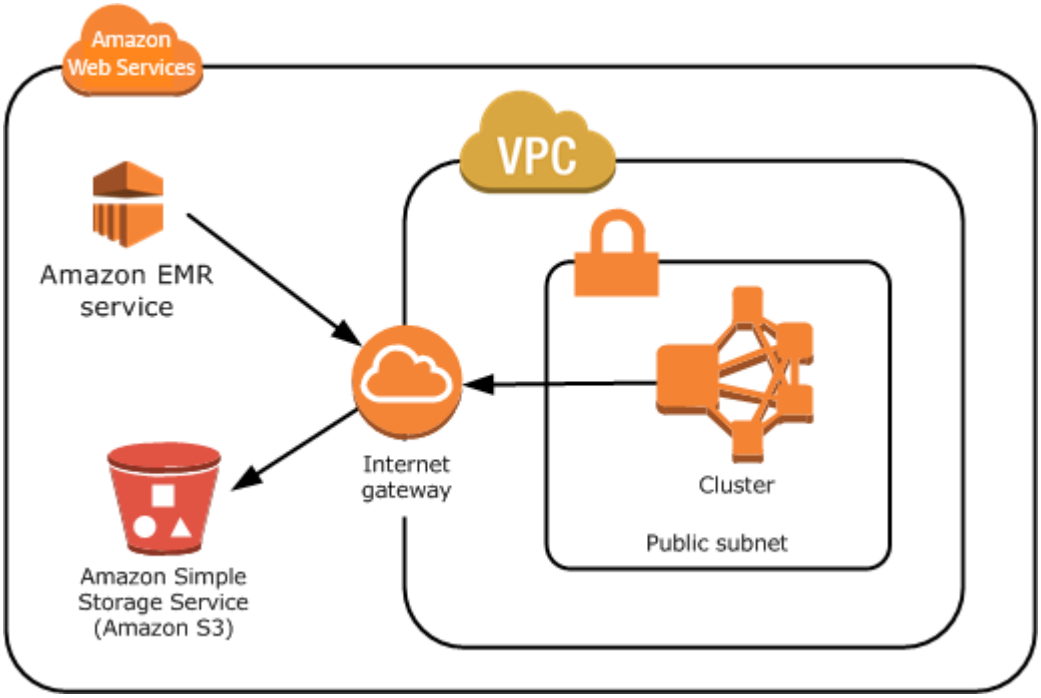
EMR clusters in a public subnet require a connected internet gateway. This is because Amazon EMR clusters must access AWS services and Amazon EMR. If a service, such as Amazon S3, provides the ability to create a VPC endpoint, you can access those services using the endpoint instead of accessing a public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

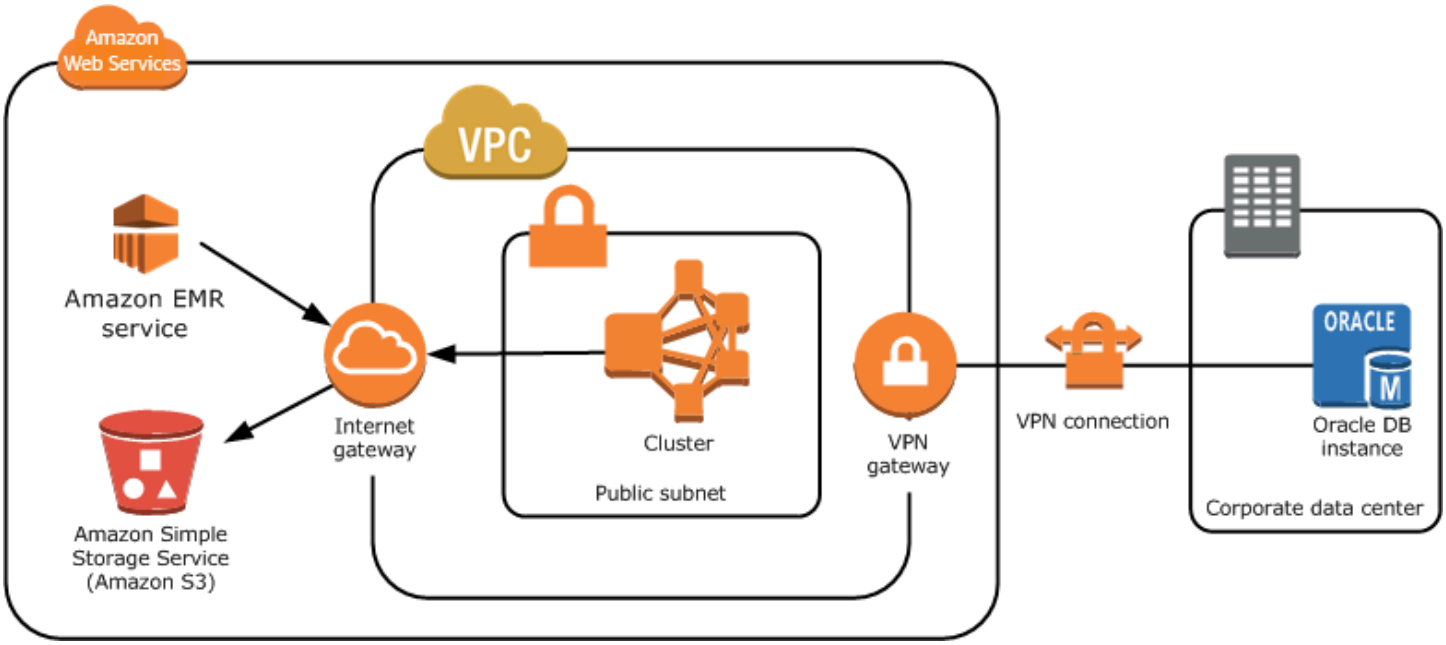
If you have additional AWS resources that you do not want connected to the internet gateway, you can launch those components in a private subnet that you create within your VPC.

Clusters running in a public subnet use two security groups: one for the primary node and another for core and task nodes. For more information, see Control network traffic with security groups for your Amazon EMR cluster.

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:

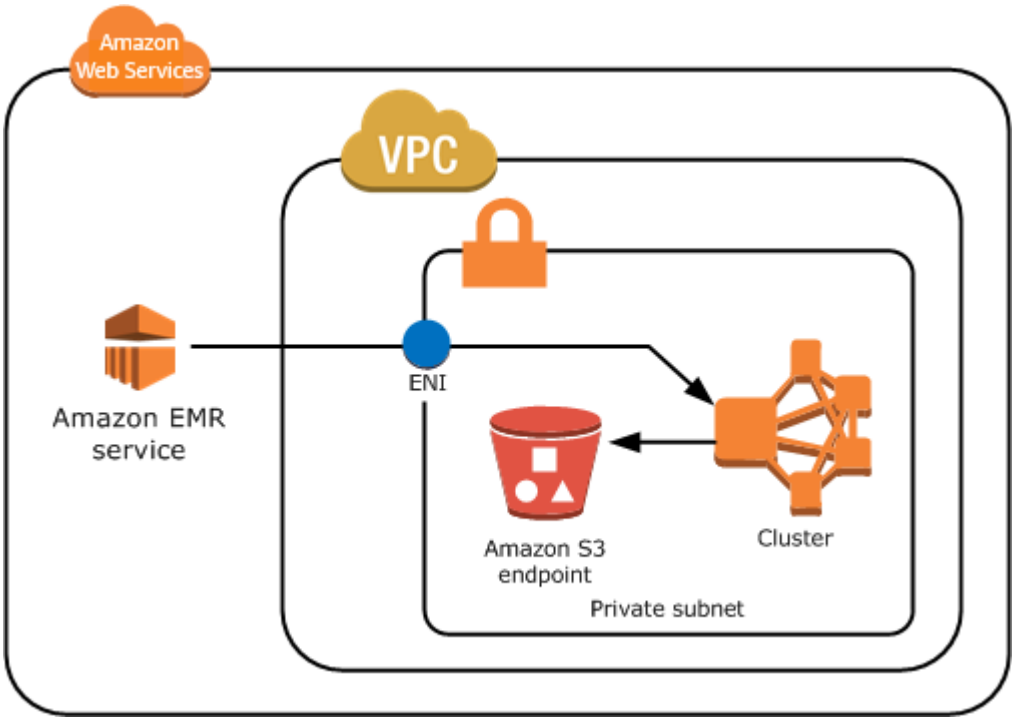
- To access AWS services that do not provide a VPC endpoint, you still must use a NAT instance or an internet gateway.
- At a minimum, you must provide a route to the Amazon EMR service logs bucket and Amazon Linux repository in Amazon S3. For more information, see Sample policies for private subnets that access Amazon S3

- 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.

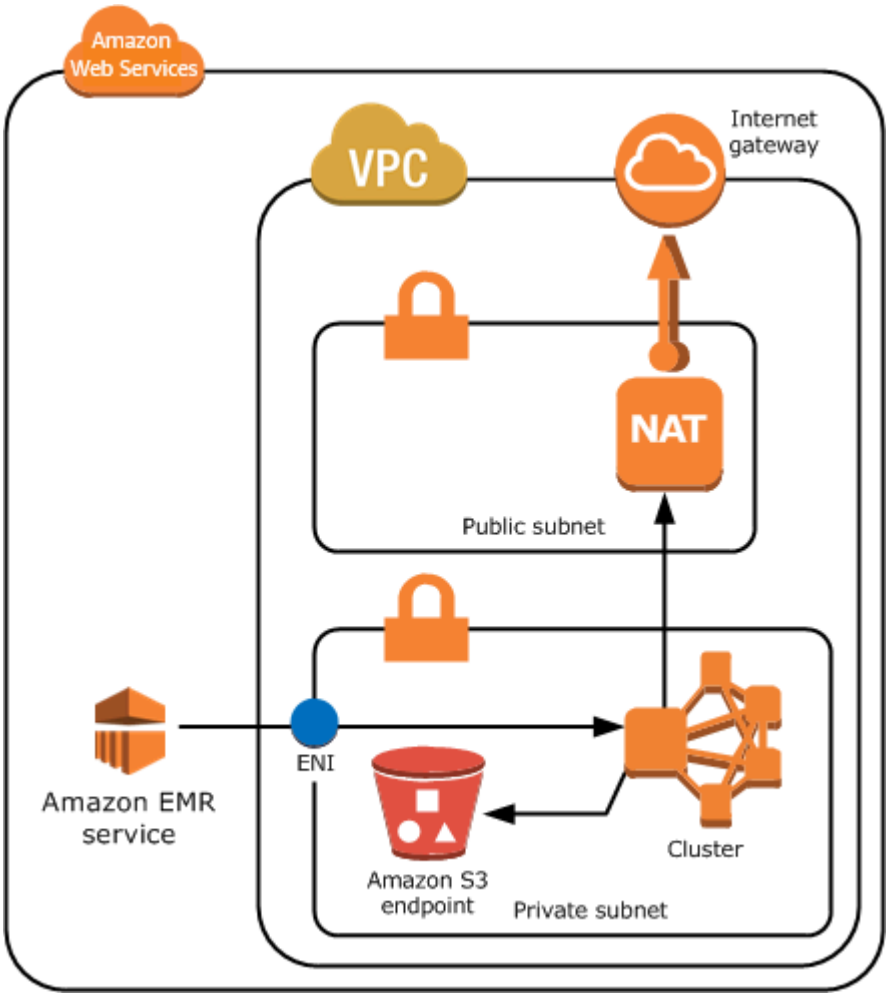
Amazon EMR creates and uses different default security groups for the clusters in a private subnet: ElasticMapReduce-Master-Private, ElasticMapReduce-Slave-Private, and ElasticMapReduce-ServiceAccess. For more information, see Control network traffic with security groups for your Amazon EMR cluster.

For a complete listing of NACLs of your cluster, choose **Security groups for Primary** 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 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 an IAM role to administer subnets, and a second role 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*.

Set up a VPC to host Amazon EMR 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 with Amazon EMR

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 for your Amazon EMR cluster.

To manage the cluster on a VPC, Amazon EMR attaches a network device to the primary 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.

Console

To launch a cluster into a VPC with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose **Create cluster**.
3. Under **Networking**, go to the **Virtual private cloud (VPC)** field. Enter the name of your VPC or choose **Browse** to select your VPC. Alternatively, choose **Create VPC** to create a VPC that you can use for your cluster.
4. Choose any other options that apply to your cluster.
5. To launch your cluster, choose **Create cluster**.

AWS CLI

To launch a cluster into a VPC with the AWS CLI

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

After your VPC is configured, you can launch 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 primary 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.

Ensuring available IP addresses for an EMR cluster on EC2

To ensure that a subnet with enough free IP addresses is available when you launch, the EC2 subnet selection checks IP availability. It The creation process uses a subnet with the necessary count of IP address to launch core, primary and task nodes as required, even if upon initial creation, only core nodes for the cluster are created. EMR checks the number of IP addresses required to launch primary and task nodes during creation, as well as calculating separately the number of IP addresses needed to launch core nodes. The minimum number of primary and task instances or nodes required is determined automatically by Amazon EMR.

> **Important**
>
> If no subnets in the VPC have enough available IPs to accommodate essential nodes, an error is returned and the cluster isn't created.

In most deployment cases, there is a time difference between each launch of core, primary and task nodes. Additionally, it's possible for multiple clusters to share a subnet. In these cases, IP-address availability can fluctuate and subsequent task-node launches, for instance, can be limited by available IP addresses.

Sample policies for private subnets that access Amazon S3

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 the following bucket:

```
aws157-logs-${AWS::Region}/*
```

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. The following example policy provides permissions to access Amazon Linux repositories and the Amazon EMR system bucket for collecting Spark event logs. It shows a few sample resource names for the buckets.

For more information about using IAM policies with Amazon VPC endpoints, see Endpoint policies for Amazon S3.

The following policy example contains sample resources in the us-east-1 region.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AmazonLinuxAMIRepositoryAccess",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::packages.us-east-1.amazonaws.com/*",
        "arn:aws:s3:::repo.us-east-1.amazonaws.com/*"
      ]
    },
    {
      "Sid": "EnableApplicationHistory",
      "Effect": "Allow",
      "Action": [
        "s3:Put*",
        "s3:Get*",
        "s3:Create*",
```

```
        "s3:Abort*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::prod.us-east-1.appinfo.src/*"
      ]
    }
  ]
}
```

The following example policy provides the permissions required to access Amazon Linux 2 repositories in the us-east-1 region.

```
{
   "Statement": [
       {
           "Sid": "AmazonLinux2AMIRepositoryAccess",
           "Effect": "Allow",
           "Principal": "*",
           "Action": "s3:GetObject",
           "Resource": [
            "arn:aws:s3:::amazonlinux.us-east-1.amazonaws.com/*",
            "arn:aws:s3:::amazonlinux-2-repos-us-east-1/*"
           ]
       }
   ]
}
```

The following example policy provides the permissions required to access Amazon Linux 2023 repositories in the us-east-1 region.

```
{
    "Statement": [
        {
            "Sid": "AmazonLinux2023AMIRepositoryAccess",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": [
                 "arn:aws:s3:::al2023-repos-us-east-1-de612dc2/*"
            ]
        }
```

```
    ]
}
```

Available regions

The following table contains a list of buckets by region, and includes both an Amazon Resource Name (ARN) for the respository and a string that represents the ARN for the `appinfo.src`. The ARN, or Amazon Resource Name, is a string that uniquely identifies an AWS resource.

Region	Repository buckets	AppInfo bucket
US East (Ohio)	"arn:aws:s3:::packages.us-east-2.amazonaws.com/","arn:aws:s3:::repo.us-east-2.amazonaws.com/","arn:aws:s3:::repo.us-east-2.emr.amazonaws.com/*"	"arn:aws:s3:::prod.us-east-2.appinfo.src/*"
US East (N. Virginia)	"arn:aws:s3:::packages.us-east-1.amazonaws.com/","arn:aws:s3:::repo.us-east-1.amazonaws.com/","arn:aws:s3:::repo.us-east-1.emr.amazonaws.com/*"	"arn:aws:s3:::prod.us-east-1.appinfo.src/*"
US West (N. California)	"arn:aws:s3:::packages.us-west-1.amazonaws.com/","arn:aws:s3:::repo.us-west-1.amazonaws.com/","arn:aws:s3:::repo.us-west-1.emr.amazonaws.com/*"	"arn:aws:s3:::prod.us-west-1.appinfo.src/*"
US West (Oregon)	"arn:aws:s3:::packages.us-west-2.amazonaws.com/","arn:aws:s3:::repo.us-west-2.amazonaws.com/","arn:aws:s3:::repo.us-west-2.emr.amazonaws.com/*"	"arn:aws:s3:::prod.us-west-2.appinfo.src/*"
Africa (Cape Town)	"arn:aws:s3:::packages.af-south-1.amazonaws.com/","arn:aws:s3:::repo.af-south-1.amazonaws.com/","ar	"arn:aws:s3:::prod.af-south-1.appinfo.src/*"

Region	Repository buckets	AppInfo bucket
	n:aws:s3:::repo.af-south-1.emr.amazonaws.com/*"	
Africa (Cape Town)	"arn:aws:s3:::packages.ap-east-1.amazonaws.com/","arn:aws:s3:::repo.ap-east-1.amazonaws.com/","arn:aws:s3:::repo.ap-east-1.emr.amazonaws.com/*"	"arn:aws:s3:::prod.ap-east-1.appinfo.src/*"
Asia Pacific (Hyderabad)	"arn:aws:s3:::packages.ap-south-2.amazonaws.com/","arn:aws:s3:::repo.ap-south-2.amazonaws.com/","arn:aws:s3:::repo.ap-south-2.emr.amazonaws.com/*"	"arn:aws:s3:::prod.ap-south-2.appinfo.src/*"
Asia Pacific (Jakarta)	"arn:aws:s3:::packages.ap-southeast-3.amazonaws.com/","arn:aws:s3:::repo.ap-southeast-3.amazonaws.com/","arn:aws:s3:::repo.ap-southeast-3.emr.amazonaws.com/*"	"arn:aws:s3:::prod.ap-southeast-3.appinfo.src/*"
Asia Pacific (Malaysia)	"arn:aws:s3:::packages.ap-southeast-5.amazonaws.com/","arn:aws:s3:::repo.ap-southeast-5.amazonaws.com/","arn:aws:s3:::repo.ap-southeast-5.emr.amazonaws.com/*"	"arn:aws:s3:::prod.ap-southeast-5.appinfo.src/*"
Asia Pacific (Melbourne)	"arn:aws:s3:::packages.ap-southeast-4.amazonaws.com/","arn:aws:s3:::repo.ap-southeast-4.amazonaws.com/","arn:aws:s3:::repo.ap-southeast-4.emr.amazonaws.com/*"	"arn:aws:s3:::prod.ap-south-4.appinfo.src/*"

Region	Repository buckets	AppInfo bucket
Asia Pacific (Mumbai)	"arn:aws:s3:::packages.ap-south-1.amazonaws.com/","arn:aws:s3:::repo.ap-south-1.amazonaws.com/","arn:aws:s3:::repo.ap-south-1.emr.amazonaws.com/*"	"arn:aws:s3:::prod.ap-south-1.appinfo.src/*"
Asia Pacific (Osaka)	"arn:aws:s3:::packages.ap-northeast-3.amazonaws.com/","arn:aws:s3:::repo.ap-northeast-3.amazonaws.com/","arn:aws:s3:::repo.ap-northeast-3.emr.amazonaws.com/*"	"arn:aws:s3:::prod.ap-northeast-3.appinfo.src/*"
Asia Pacific (Seoul)	"arn:aws:s3:::packages.ap-northeast-2.amazonaws.com/","arn:aws:s3:::repo.ap-northeast-2.amazonaws.com/","arn:aws:s3:::repo.ap-northeast-2.emr.amazonaws.com/*"	"arn:aws:s3:::prod.ap-northeast-2.appinfo.src/*"
Asia Pacific (Singapore)	"arn:aws:s3:::packages.ap-southeast-1.amazonaws.com/","arn:aws:s3:::repo.ap-southeast-1.amazonaws.com/","arn:aws:s3:::repo.ap-southeast-1.emr.amazonaws.com/*"	"arn:aws:s3:::prod.ap-southeast-1.appinfo.src/*"
Asia Pacific (Sydney)	"arn:aws:s3:::packages.ap-southeast-2.amazonaws.com/","arn:aws:s3:::repo.ap-southeast-2.amazonaws.com/","arn:aws:s3:::repo.ap-southeast-2.emr.amazonaws.com/*"	"arn:aws:s3:::prod.ap-southeast-2.appinfo.src/*"
Asia Pacific (Tokyo)	"arn:aws:s3:::packages.ap-northeast-1.amazonaws.com/","arn:aws:s3:::repo.ap-northeast-1.amazonaws.com/","arn:aws:s3:::repo.ap-northeast-1.emr.amazonaws.com/*"	"arn:aws:s3:::prod.ap-northeast-1.appinfo.src/*"

Region	Repository buckets	AppInfo bucket
Canada (Central)	"arn:aws:s3:::packages.ca-central-1.amazonaws.com/","arn:aws:s3:::repo.ca-central-1.amazonaws.com/","arn:aws:s3:::repo.ca-central-1.emr.amazonaws.com/*"	"arn:aws:s3:::prod.ca-central-1.appinfo.src/*"
Canada West (Calgary)	"arn:aws:s3:::packages.ca-west-1.amazonaws.com/","arn:aws:s3:::repo.ca-west-1.amazonaws.com/","arn:aws:s3:::repo.ca-west-1.emr.amazonaws.com/*"	"arn:aws:s3:::prod.ca-west-1.appinfo.src/*"
Europe (Frankfurt)	"arn:aws:s3:::packages.eu-central-1.amazonaws.com/","arn:aws:s3:::repo.eu-central-1.amazonaws.com/","arn:aws:s3:::repo.eu-central-1.emr.amazonaws.com/*"	"arn:aws:s3:::prod.eu-central-1.appinfo.src/*"
Europe (Ireland)	"arn:aws:s3:::packages.eu-west-1.amazonaws.com/","arn:aws:s3:::repo.eu-west-1.amazonaws.com/","arn:aws:s3:::repo.eu-west-1.emr.amazonaws.com/*"	"arn:aws:s3:::prod.eu-west-1.appinfo.src/*"
Europe (London)	"arn:aws:s3:::packages.eu-west-2.amazonaws.com/","arn:aws:s3:::repo.eu-west-2.amazonaws.com/","arn:aws:s3:::repo.eu-west-2.emr.amazonaws.com/*"	"arn:aws:s3:::prod.eu-west-2.appinfo.src/*"
Europe (Milan)	"arn:aws:s3:::packages.eu-south-1.amazonaws.com/","arn:aws:s3:::repo.eu-south-1.amazonaws.com/","arn:aws:s3:::repo.eu-south-1.emr.amazonaws.com/*"	"arn:aws:s3:::prod.eu-south-1.appinfo.src/*"

Region	Repository buckets	AppInfo bucket
Europe (Paris)	"arn:aws:s3:::packages.eu-west-3.amazonaws.com/","arn:aws:s3:::repo.eu-west-3.amazonaws.com/","arn:aws:s3:::repo.eu-west-3.emr.amazonaws.com/*"	"arn:aws:s3:::prod.eu-west-3.appinfo.src/*"
Europe (Spain)	"arn:aws:s3:::packages.eu-south-2.amazonaws.com/","arn:aws:s3:::repo.eu-south-2.amazonaws.com/","arn:aws:s3:::repo.eu-south-2.emr.amazonaws.com/*"	"arn:aws:s3:::prod.eu-south-2.appinfo.src/*"
Europe (Stockholm)	"arn:aws:s3:::packages.eu-north-1.amazonaws.com/","arn:aws:s3:::repo.eu-north-1.amazonaws.com/","arn:aws:s3:::repo.eu-north-1.emr.amazonaws.com/*"	"arn:aws:s3:::prod.eu-north-1.appinfo.src/*"
Europe (Zurich)	"arn:aws:s3:::packages.eu-central-2.amazonaws.com/","arn:aws:s3:::repo.eu-central-2.amazonaws.com/","arn:aws:s3:::repo.eu-central-2.emr.amazonaws.com/*"	"arn:aws:s3:::prod.eu-central-2.appinfo.src/*"
Israel (Tel Aviv)	"arn:aws:s3:::packages.il-central-1.amazonaws.com/","arn:aws:s3:::repo.il-central-1.amazonaws.com/","arn:aws:s3:::repo.il-central-1.emr.amazonaws.com/*"	"arn:aws:s3:::prod.il-central-1.appinfo.src/*"

Region	Repository buckets	AppInfo bucket
Middle East (Bahrain)	"arn:aws:s3:::packages.me-south-1.amazonaws.com/","arn:aws:s3:::repo.me-south-1.amazonaws.com/","arn:aws:s3:::repo.me-south-1.emr.amazonaws.com/*"	"arn:aws:s3:::prod.me-south-1.appinfo.src/*"
Middle East (UAE)	"arn:aws:s3:::packages.me-central-1.amazonaws.com/","arn:aws:s3:::repo.me-central-1.amazonaws.com/","arn:aws:s3:::repo.me-central-1.emr.amazonaws.com/*"	"arn:aws:s3:::prod.me-central-1.appinfo.src/*"
South America (São Paulo)	"arn:aws:s3:::packages.sa-east-1.amazonaws.com/","arn:aws:s3:::repo.sa-east-1.amazonaws.com/","arn:aws:s3:::repo.sa-east-1.emr.amazonaws.com/*"	"arn:aws:s3:::prod.sa-east-1.appinfo.src/*"
AWS GovCloud (US-East)	"arn:aws:s3:::packages.us-gov-east-1.amazonaws.com/","arn:aws:s3:::repo.us-gov-east-1.amazonaws.com/","arn:aws:s3:::repo.us-gov-east-1.emr.amazonaws.com/*"	"arn:aws:s3:::prod.us-gov-east-1.appinfo.src/*"
AWS GovCloud (US-West)	"arn:aws:s3:::packages.us-gov-west-1.amazonaws.com/","arn:aws:s3:::repo.us-gov-west-1.amazonaws.com/","arn:aws:s3:::repo.us-gov-west-1.emr.amazonaws.com/*"	"arn:aws:s3:::prod.me-south-1.appinfo.src/*"

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 an Amazon EMR cluster with instance fleets or uniform instance groups

When you create a cluster and specify the configuration of the primary 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 new Amazon EMR console, you can choose to use either instance groups or instance fleets when you create a cluster, and you have the option to use Spot Instances with each. With the old Amazon EMR console, if you use the default **Quick Options** settings when you create your 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**.

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 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 primary 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).

For more information about configuring instance fleets, see Planning and configuring instance fleets for your Amazon EMR cluster.

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 primary 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 Use Amazon EMR cluster scaling to adjust for changing workloads.

For more information about configuring uniform instance groups, see Configure uniform instance groups for your Amazon EMR cluster.

Working with instance fleets and instance groups

Topics

- Planning and configuring instance fleets for your Amazon EMR cluster

- Reconfiguring instance fleets for your Amazon EMR cluster
- Use capacity reservations with instance fleets in Amazon EMR
- Configure uniform instance groups for your Amazon EMR cluster
- Availability Zone flexibility for an Amazon EMR cluster
- Configuring Amazon EMR cluster instance types and best practices for Spot instances

Planning and configuring instance fleets for your Amazon EMR cluster

> **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 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 for working with instance fleets

Consider the following items when you use instance fleets with Amazon EMR.

- You can have one instance fleet, and only one, per node type (primary, 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).
- 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 primary 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 specify one of the following allocation strategies for your Spot Instances: price-capacity optimized, capacity-optimized, capacity-optimized-prioritized, lowest-price, or diversified across all pools.
- For each fleet, you can apply the following allocation strategies for your On-Demand Instances: the lowest-price strategy or the prioritized strategy.
- For each fleet with On-Demand Instances, you can choose to apply capacity reservation options.
- If you use allocation strategy for instance fleets, the following considerations apply when you choose subnets for your EMR cluster:
  - When Amazon EMR provisions a cluster with a task fleet, it filters out subnets that lack enough available IP addresses to provision all instances of the requested EMR cluster. This includes

IP addresses required for the primary, core, and task instance fleets during cluster launch. Amazon EMR then leverages its allocation strategy to determine the instance pool, based on instance type and remaining subnets with sufficient IP addresses, to launch the cluster.

- If Amazon EMR cannot launch the whole cluster due to insufficient available IP addresses, it will attempt to identify subnets with enough free IP addresses to launch the essential (core and primary) instance fleets. In such scenarios, your task instance fleet will go into a suspended state, rather than terminating the cluster with an error.
- If none of the specified subnets contain enough IP addresses to provision the essential core and primary instance fleets, the cluster launch will fail with a **VALIDATION_ERROR**. This triggers a **CRITICAL** severity cluster termination event, notifying you that the cluster cannot be launched. To prevent this issue, we recommend increasing the number of IP addresses in your subnets.

- If you run Amazon EMR release **emr-7.7.0** and above, and you use allocation strategy for instance fleets, you can scale the cluster up to 4000 EC2 instances and 14000 EBS volumes per instance fleet. For release versions below **emr-7.7.0**, the cluster can be scaled up only to 2000 EC2 instances and 7000 EBS volumes per instance fleet.
- When you launch On-Demand Instances, you can use open or targeted capacity reservations for primary, core, and task nodes in your accounts. You might see insufficient capacity with On-Demand Instances with allocation strategy for instance fleets. We recommend that you specify multiple instance types to diversify and reduce the chance of experiencing insufficient capacity. For more information, see the section called “Use capacity reservations with instance fleets in Amazon EMR”.

Instance fleet options

Use the following guidelines to understand instance fleet options.

Topics

- Setting target capacities
- Launch options
- Multiple subnet (Availability Zones) options
- Master node configuration

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 primary 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 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.

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.

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 old AWS Management Console or using the API `RunJobFlow`. Note that you can't customize allocation strategy in the new console. Using the allocation strategy option requires additional service role permissions. If you use the default Amazon EMR service role and managed policy (`EMR_DefaultRole` 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).

For more information about Spot Instances, see Spot Instances in the Amazon EC2 User Guide. For more information about On-Demand Instances, see On-Demand Instances in the Amazon EC2 User Guide.

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 Allocation strategy permissions. Note that you can't customize capacity reservations in the new console.

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 primary 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 primary 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 primary 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 primary 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, Amazon EMR API, or Amazon EMR console with an allocation strategy, you can specify up to 30 Amazon EC2 instance types per fleet. With the default Amazon EMR cluster instance fleet configuration, you can have up to 5 instance types per fleet. We recommend that you use the allocation strategy option for faster cluster provisioning, more accurate Spot Instance allocation, and fewer Spot Instance interruptions.

Topics

- Allocation strategy with On-Demand Instances
- Allocation strategy with Spot Instances
- Allocation strategy permissions
- Required IAM permissions for an allocation strategy

Allocation strategy with On-Demand Instances

The following allocation strategies are available for your On-Demand Instances:

`lowest-price` **(default)**

The lowest-price allocation strategy launches On-Demand instances from the lowest priced pool that has available capacity. If the lowest-priced pool doesn't have available capacity, the On-Demand Instances come from the next lowest-priced pool with available capacity.

`prioritized`

The prioritized allocation strategy lets you specify a priority value for each instance type for your instance fleet. Amazon EMR launches your On-Demand Instances that have the highest priority. If you use this strategy, you must configure the priority for at least one instance type. If you don't configure the priority value for an instance type, Amazon EMR assigns the lowest priority to that instance type. Each instance fleet (primary, core, or task) in a cluster can have a different priority value for a given instance type.

> **Note**
>
> If you use the **capacity-optimized-prioritized** Spot allocation strategy, Amazon EMR applies the same priorities to both your On-Demand Instances and Spot instances when you set priorities.

Allocation strategy with Spot Instances

For *Spot Instances* you can choose from one of the following allocation strategies:

`price-capacity-optimized` (recommended)

The price-capacity optimized allocation strategy launches Spot instances from the Spot instance pools that have the highest capacity available and the lowest price for the number of instances that are launching. As a result, the price-capacity optimized strategy typically has a higher chance of getting Spot capacity, and delivers lower interruption rates.This is the default strategy for Amazon EMR releases 6.10.0 and higher.

`capacity-optimized`

The capacity-optimized allocation strategy launches Spot Instances into the most available pools with the lowest chance of interruption in the near term. This is a good option for workloads that might have a higher cost of interruption associated with work that gets restarted. This is the default strategy for Amazon EMR releases 6.9.0 and lower.

`capacity-optimized-prioritized`

The capacity-optimized-prioritized allocation strategy lets you specify a priority value for each instance type in your instance fleet. Amazon EMR optimizes for capacity first, but it respects instance type priorities on a best-effort basis, such as if the priority doesn't significantly

affect the fleet's ability to provision optimal capacity. We recommend this option if you have workloads that must have a minimal amount of disruption that still have a need for certain instance types. If you use this strategy, you must configure the priority for at least one instance type. If you don't configure a priority for any instance type, Amazon EMR assigns the lowest priority value to that instance type. Each instance fleet (primary, core, or task) in a cluster can have a different priority value for a given instance type.

> **Note**
>
> If you use the **prioritized** On-Demand allocation strategy, Amazon EMR applies the same priority value to both your On-Demand and Spot Instances when you set priorities.

`diversified`

With the diversified allocation strategy, Amazon EC2 distributes Spot Instances across all Spot capacity pools.

`lowest-price`

The lowest-price allocation strategy launches Spot Instances from the lowest priced pool that has available capacity. If the lowest-priced pool doesn't have available capacity, the Spot Instances come from the next lowest priced pool that has available capacity. If a pool runs out of capacity before it fulfills your requested capacity, the Amazon EC2 fleet draws from the next lowest priced pool to continue to fulfill your request. To ensure that your desired capacity is met, you might receive Spot Instances from several pools. Because this strategy only considers instance price, and does not consider capacity availability, it might lead to high interruption rates.

Allocation strategy permissions

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 Allocation strategy permissions.

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 in Amazon EMR.

Required IAM permissions for an allocation strategy

Your Service role for Amazon EMR (EMR role) requires additional permissions to create a cluster that uses the allocation strategy option for On-Demand or Spot Instance fleets.

We automatically include these permissions in the default Amazon EMR service role `EMR_DefaultRole` and the Amazon EMR managed policy `AmazonEMRServicePolicy_v2`.

If you use a custom service role or managed policy for your cluster, you must add the following permissions:

JSON

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

```
        "ec2:DeleteLaunchTemplate",
        "ec2:CreateLaunchTemplate",
        "ec2:DescribeLaunchTemplates",
        "ec2:CreateLaunchTemplateVersion",
        "ec2:CreateFleet"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowEC2Deletelaunchtemplate"
    }
  ]
}
```

The following service role permissions are 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.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeCapacityReservations",
        "ec2:DescribeLaunchTemplateVersions",
        "ec2:DeleteLaunchTemplateVersions"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowEC2Describecapacityreservations"
    }
  ]
}
```

Example

To use targeted capacity reservations, you must include the following additional permissions.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeCapacityReservations",
        "ec2:DescribeLaunchTemplateVersions",
        "ec2:DeleteLaunchTemplateVersions",
        "resource-groups:ListGroupResources"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowEC2Describecapacityreservations"
    }
  ]
}
```

Configure instance fleets for your cluster

Console

To create a cluster with instance fleets with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and choose **Create cluster**.
3. Under **Cluster configuration**, choose **Instance fleets**.
4. For each **Node group**, select **Add instance type** and choose up to 5 instance types for primary and core instance fleets and up to fifteen instance types for task instance fleets. Amazon EMR might provision any mix of these instance types when it launches the cluster.

5. Under each node group type, choose the **Actions** dropdown menu next to each instance to change these settings:

   Add EBS volumes

   Specify EBS volumes to attach to the instance type after Amazon EMR provisions it.

   Edit weighted capacity

   For the core node group, change this value to any number of units that fits your applications. The number of YARN vCores for each fleet instance type is used as the default weighted capacity units. You can't edit weighted capacity for the primary node.

   Edit maximum Spot price

   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. If the current Spot price in an Availability Zone is below your maximum Spot price, Amazon EMR provisions Spot Instances. You pay the Spot price, not necessarily the maximum Spot price.

6. Optionally, to add security groups for your nodes, expand **EC2 security groups (firewall)** in the **Networking** section and select your security group for each node type.
7. Optionally, select the check box next to **Apply allocation strategy** if you want to use the allocation strategy option, and select the allocation strategy that you want to specify for the Spot Instances. You shouldn't select this option if your Amazon EMR service role doesn't have the required permissions. For more information, see Allocation strategy for instance fleets.
8. Choose any other options that apply to your cluster.
9. To launch your cluster, choose **Create cluster**.

AWS CLI

To create and launch a cluster with instance fleets with the AWS CLI, follow these guidelines:

- 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 Examples: Creating a cluster with the instance fleets configuration.
- 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 Examples: Creating a cluster with the instance fleets configuration.
- 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.

Examples: Creating 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 primary, 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.xlarg
 \

 InstanceFleetType=CORE,TargetOnDemandCapacity=1,InstanceTypeConfigs=['{InstanceType=m5.xlarge}
```

Example Example: Spot primary, 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 primary, 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 primary, 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
```

Example Example: On-Demand primary, 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.xlarg
 \
    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
 \
    InstanceFleetType=TASK,TargetOnDemandCapacity=3,TargetSpotCapacity=3,\
InstanceTypeConfigs=['{InstanceType=m5.xlarge,BidPrice=0.5,WeightedCapacity=3}']
```

Example Example: Spot primary, 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_CLU
\
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 Example: Multiple custom AMIs, multiple instance types, on-demand primary, 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 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 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 primary, 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 primary 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 primary 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_CLU
\
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 primary 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"
        }
    ]
}
```

Reconfiguring instance fleets for your Amazon EMR cluster

With Amazon EMR version 5.21.0 and later, you can reconfigure cluster applications and specify additional configuration classifications for each instance fleet in a running cluster. To do so, you can use the AWS Command Line Interface (AWS CLI), or the AWS SDK.

You can track the state of an instance fleet, by viewing the CloudWatch events. For more information, see Instance fleet reconfiguration events.

> **Note**
>
> You can only override the cluster Configurations object specified during cluster creation. For more information about Configurations objects, see RunJobFlow request syntax. If there are differences between the existing configuration and the file that you supply, Amazon EMR resets manually modified configurations, such as configurations that you have modified while connected to your cluster using SSH, to the cluster defaults for the specified instance fleet.

When you submit a reconfiguration request using the Amazon EMR console, the AWS Command Line interface (AWS CLI), or the AWS SDK, Amazon EMR checks the existing on-cluster configuration file. If there are differences between the existing configuration and the file that you supply, Amazon EMR initiates reconfiguration actions, restarts some applications, and resets any manually modified configurations, such as configurations that you have modified while connected to your cluster using SSH, to the cluster defaults for the specified instance fleet.

Reconfiguration behaviors

Reconfiguration overwrites on-cluster configuration with the newly submitted configuration set, and can overwrite configuration changes made outside of the reconfiguration API.

Amazon EMR follows a rolling process to reconfigure instances in the Task and Core instance fleet. Only a percentage of the instances for a single instance type are modified and restarted at a time. If your instance fleet has multiple different instance type configurations, they would reconfigure in parallel.

Reconfigurations are declared at the InstanceTypeConfig level. For a visual example, refer to Reconfigure an instance fleet. You can submit reconfiguration requests that contain updated configuration settings for one or more instance types within a single request. You must include all instance types that are part of your instance fleet in the modify request; however, instance types with populated configuration fields will undergo reconfiguration, while other `InstanceTypeConfig` instances in the fleet remain unchanged. A reconfiguration is considered successful only when all instances of the specified instance types complete reconfiguration. If any instance fails to reconfigure, the entire Instance Fleet automatically reverts to its last known stable configuration.

Limitations

When you reconfigure an instance fleet in a running cluster, consider the following limitations:

- Non-YARN applications can fail during restart or cause cluster issues, especially if the applications aren't configured properly. Clusters approaching maximum memory and CPU usage may run into issues after the restart process. This is especially true for the primary instance fleet. Consult the Troubleshoot instance fleet reconfiguration section.
- Resizes and Reconfiguration operation do not happen in parallel. Reconfiguration requests will wait for an ongoing resize and vice versa.
- Resizes and Reconfiguration operation do not happen in parallel. Reconfiguration requests will wait for an ongoing resize and vice versa.
- After reconfiguring an instance fleet, Amazon EMR restarts the applications to allow the new configurations to take effect. Job failure or other unexpected application behavior might occur if the applications are in use during reconfiguration.
- If a reconfiguration for any instance type config under an instance fleet fails, Amazon EMR reverses the configuration parameters to the previous working version for the entire instance fleet, along with emitting events and updating state details. If the reversion process fails too, you must submit a new `ModifyInstanceFleet` request to recover the instance fleet from the `ARRESTED` state. Reversion failures result in Instance fleet reconfiguration events and state change.
- Reconfiguration requests for Phoenix configuration classifications are only supported in Amazon EMR version 5.23.0 and later, and are not supported in Amazon EMR version 5.21.0 or 5.22.0.
- Reconfiguration requests for HBase configuration classifications are only supported in Amazon EMR version 5.30.0 and later, and are not supported in Amazon EMR versions 5.23.0 through 5.29.0.
- Reconfiguring hdfs-encryption-zones classification or any of the Hadoop KMS configuration classifications is not supported on an Amazon EMR cluster with multiple primary nodes.
- Amazon EMR currently doesn't support certain reconfiguration requests for the YARN capacity scheduler that require restarting the YARN ResourceManager. For example, you cannot completely remove a queue.
- When YARN needs to restart, all running YARN jobs are typically terminated and lost. This might cause data processing delays. To run YARN jobs during a YARN restart, you can either create an Amazon EMR cluster with multiple primary nodes or set yarn.resourcemanager.recovery.enabled

to `true` in your yarn-site configuration classification. For more information about using multiple master nodes, see High availability YARN ResourceManager.

Reconfigure an instance fleet

Using the AWS CLI

Use the `modify-instance-fleet` command to specify a new configuration for an instance fleet in a running cluster.

> **Note**
>
> In the following examples, replace **j-2AL4XXXXXX5T9** with your cluster ID, and replace **if-1xxxxxxx9** with your instance fleet ID.

Example – Replace a configuration for an instance fleet

> **Warning**
>
> Specify all `InstanceTypeConfig` fields that you used at launch. Not including fields can result in overwriting specifications you declared at launch. Refer to InstanceTypeConfig for a list.

The following example references a configuration JSON file called instanceFleet.json to edit the property of the YARN NodeManager disk health checker for an instance fleet.

Instance Fleet Modification JSON

1. Prepare your configuration classification, and save it as instanceFleet.json in the same directory where you will run the command.

```
{
    "InstanceFleetId":"if-1xxxxxxx9",
    "InstanceTypeConfigs": [
            {
                "InstanceType": "m5.xlarge",
               other InstanceTypeConfig fields
                "Configurations": [
```

```
                    {
                        "Classification": "yarn-site",
                        "Properties": {
                            "yarn.nodemanager.disk-health-checker.enable":"true",
                            "yarn.nodemanager.disk-health-checker.max-disk-
utilization-per-disk-percentage":"100.0"
                        }
                    }
                ]
            },
            {
                "InstanceType": "r5.xlarge",
               other InstanceTypeConfig fields
                "Configurations": [
                    {
                        "Classification": "yarn-site",
                        "Properties": {
                            "yarn.nodemanager.disk-health-checker.enable":"false",
                            "yarn.nodemanager.disk-health-checker.max-disk-
utilization-per-disk-percentage":"70.0"
                        }
                    }
                ]
            }
        ]
```

2. Run the following command.

```
aws emr modify-instance-fleet \
--cluster-id j-2AL4XXXXXX5T9 \
--region us-west-2 \
--instance-fleet instanceFleet.json
```

Example – Add a configuration to an instance fleet

If you want to add a configuration to an instance type, you must include all previously specified configurations for that instance type in your new `ModifyInstanceFleet` request. Otherwise, the previously specified configurations are removed.

The following example adds a property for the YARN NodeManager virtual memory checker. The configuration also includes previously specified values for the YARN NodeManager disk health checker so that the values won't be overwritten.

1. Prepare the following contents in instanceFleet.json and save it in the same directory where you will run the command.

```
{
    "InstanceFleetId":"if-1xxxxxxx9",
    "InstanceTypeConfigs": [
            {
                "InstanceType": "m5.xlarge",
                other InstanceTypeConfig fields
                "Configurations": [
                    {
                        "Classification": "yarn-site",
                        "Properties": {
                            "yarn.nodemanager.disk-health-checker.enable":"true",
                            "yarn.nodemanager.disk-health-checker.max-disk-
utilization-per-disk-percentage":"100.0",
                            "yarn.nodemanager.vmem-check-enabled":"true",
                            "yarn.nodemanager.vmem-pmem-ratio":"3.0"
                        }
                    }
                ]
            },
            {
                "InstanceType": "r5.xlarge",
                other InstanceTypeConfig fields
                "Configurations": [
                    {
                        "Classification": "yarn-site",
                        "Properties": {
                            "yarn.nodemanager.disk-health-checker.enable":"false",
                            "yarn.nodemanager.disk-health-checker.max-disk-
utilization-per-disk-percentage":"70.0"
                        }
                    }
                ]
            }
        ]
}
```

2. Run the following command.

```
aws emr modify-instance-fleet \
--cluster-id j-2AL4XXXXXX5T9 \
```

```
--region us-west-2 \
--instance-fleet instanceFleet.json
```

using the Java SDK

> **Note**
>
> In the following examples, replace **j-2AL4XXXXXX5T9** with your cluster ID, and replace **if-1xxxxxxx9** with your instance fleet ID.

The following code snippet provides a new configuration for an instance fleet using the AWS SDK for Java.

```
AWSCredentials credentials = new BasicAWSCredentials("access-key", "secret-key");
AmazonElasticMapReduce emr = new AmazonElasticMapReduceClient(credentials);

Map<String,String> hiveProperties = new HashMap<String,String>();
hiveProperties.put("hive.join.emit.interval","1000");
hiveProperties.put("hive.merge.mapfiles","true");

Configuration newConfiguration = new Configuration()
    .withClassification("hive-site")
    .withProperties(hiveProperties);

List<InstanceTypeConfig> instanceTypeConfigList = new ArrayList<>();

for (InstanceTypeConfig instanceTypeConfig : currentInstanceTypeConfigList) {
    instanceTypeConfigList.add(new InstanceTypeConfig()
        .withInstanceType(instanceTypeConfig.getInstanceType())
        .withBidPrice(instanceTypeConfig.getBidPrice())
        .withWeightedCapacity(instanceTypeConfig.getWeightedCapacity())
        .withConfigurations(newConfiguration)
    );
}

InstanceFleetModifyConfig instanceFleetModifyConfig = new
 InstanceFleetModifyConfig()
    .withInstanceFleetId("if-1xxxxxxx9")
    .withInstanceTypeConfigs(instanceTypeConfigList);
```

```
ModifyInstanceFleetRequest modifyInstanceFleetRequest = new
 ModifyInstanceFleetRequest()
    .withInstanceFleet(instanceFleetModifyConfig)
    .withClusterId("j-2AL4XXXXXX5T9");

emrClient.modifyInstanceFleet(modifyInstanceFleetRequest);
```

Troubleshoot instance fleet reconfiguration

If the reconfiguration process for any instance type within an instance fleet fails, Amazon EMR reverts the in progress reconfiguration and logs a failure message using an AAmazon CloudWatch Events events. The event provides a brief summary of the reconfiguration failure. It lists the instances for which reconfiguration has failed and corresponding failure messages. The following is an example failure message.

```
Amazon EMR couldn't revert the instance fleet if-1xxxxxxx9 in the Amazon
EMR cluster j-2AL4XXXXXX5T9 (ExampleClusterName) to the previously
successful configuration at 2021-01-01 00:00 UTC. The reconfiguration
reversion failed because of Instance i-xxxxxxx1, i-xxxxxxx2, i-xxxxxxx3
failed with message "This is an example failure message"...
```

To access node provisioning logs

Use SSH to connect to the node on which reconfiguration has failed. For instructions, see Connect to your Linux instance in the *Amazon Elastic Compute Cloud*.

Accessing logs by connecting to a node

1. Navigate to the following directory, which contains the node provisioning log files.

   ```
   /mnt/var/log/provision-node/
   ```

2. Open the reports subdirectory and search for the node provisioning report for your reconfiguration. The reports directory organizes logs by reconfiguration version number, universally unique identifier (UUID), Amazon EC2 instance IP address, and timestamp. Each report is a compressed YAML file that contains detailed information about the reconfiguration process. The following is an example report file name and path.

   ```
   /reports/2/ca598xxx-cxxx-4xxx-bxxx-6dbxxxxxxxxx/ip-10-73-xxx-
   xxx.ec2.internal/202104061715.yaml.gz
   ```

3. You can examine a report using a file viewer like zless, as in the following example.

```
zless 202104061715.yaml.gz
```

Accessing logs using Amazon S3

Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/. Open the Amazon S3 bucket that you specified when you configured the cluster to archive log files.

1. Navigate to the following folder, which contains the node provisioning log files:

```
amzn-s3-demo-bucket/elasticmapreduce/cluster id/node/instance id/provision-node/
```

2. Open the reports folder and search for the node provisioning report for your reconfiguration. The reports folder organizes logs by reconfiguration version number, universally unique identifier (UUID), Amazon EC2 instance IP address, and timestamp. Each report is a compressed YAML file that contains detailed information about the reconfiguration process. The following is an example report file name and path.

```
/reports/2/ca598xxx-cxxx-4xxx-bxxx-6dbxxxxxxxxx/ip-10-73-xxx-
xxx.ec2.internal/202104061715.yaml.gz
```

To view a log file, you can download it from Amazon S3 to your local machine as a text file. For instructions, see Downloading an object.

Each log file contains a detailed provisioning report for the associated reconfiguration. To find error message information, you can search for the `err` log level of a report. Report format depends on the version of Amazon EMR on your cluster. The following example shows error information for Amazon EMR release versions 5.32.0 and 6.2.0 and later use the following format:

```
- level: err
  message: 'Example detailed error message.'
  source: Puppet
  tags:
  - err
  time: '2021-01-01 00:00:00.000000 +00:00'
  file:
```

```
line:
```

Use capacity reservations with instance fleets in Amazon EMR

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 Allocation strategy permissions.

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`. Alternatively, 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.xlarg
 \

 InstanceFleetType=CORE,TargetOnDemandCapacity=100,InstanceTypeConfigs=['{InstanceType=c5.xlarg
{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.xlarg
 \

 InstanceFleetType=CORE,TargetOnDemandCapacity=100,InstanceTypeConfigs=['{InstanceType=c5.xlarg
{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 reservat ions	-	-	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	10	-	-

Available targeted 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 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*.

```
"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.xlarg
 \
    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.xlarg
 \

 InstanceFleetType=CORE,TargetOnDemandCapacity=100,InstanceTypeConfigs=['{InstanceType=c5.xlarg
{InstanceType=m5.xlarge},{InstanceType=r5.xlarge}'],\
LaunchSpecifications={OnDemandSpecification='{AllocationStrategy=lowest-
price,CapacityReservationOptions={CapacityReservationPreference=none}}'}
```

Where,

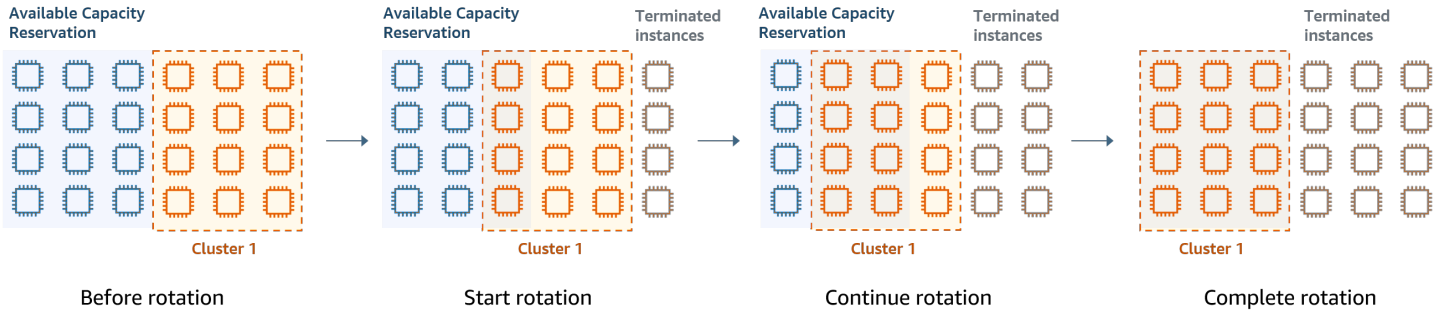
- `none-CR-cluster` is replaced with the name of your cluster that is not using any open capacity reservations.
- `subnet-22XXXX01` is replaced with the subnet ID.

Scenarios for using capacity reservations

You can benefit from using capacity reservations in the following scenarios.

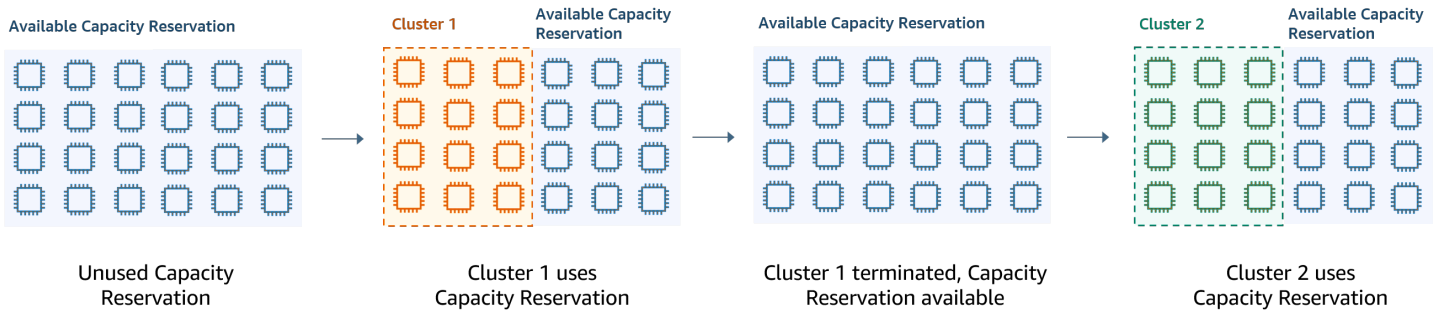
Scenario 1: Rotate a long-running cluster using capacity reservations

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 for your Amazon EMR cluster

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 primary, 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 in Amazon EMR.

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 Use Amazon EMR cluster scaling to adjust for changing workloads.

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 in Amazon EMR.

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 Amazon EMR clusters.

Use the console to configure uniform instance groups

Console

To create a cluster with instance groups with the new console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and choose **Create cluster**.
3. Under **Cluster configuration**, choose **Instance groups**.
4. Under **Node groups**, there is a section for each type of node group. For the primary node group, select the **Use multiple primary nodes** check box if you want to have 3 primary nodes. Select the **Use Spot purchasing option** check box if you want to use Spot purchasing.
5. For the primary and core node groups, select **Add instance type** and choose up to 5 instance types. For the task group, select **Add instance type** and choose up to fifteen instance types. Amazon EMR might provision any mix of these instance types when it launches the cluster.
6. Under each node group type, choose the **Actions** dropdown menu next to each instance to change these settings:

   Add EBS volumes

   Specify EBS volumes to attach to the instance type after Amazon EMR provisions it.

   Edit maximum Spot price

   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. If the current Spot price in an Availability Zone is below your maximum Spot price, Amazon EMR provisions Spot Instances. You pay the Spot price, not necessarily the maximum Spot price.
7. Optionally, expand **Node configuration** to enter a JSON configuration or to load JSON from Amazon S3.

8. Choose any other options that apply to your cluster.
9. To launch your cluster, choose **Create 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 primary, 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-7.10.0 \
  --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 primary, 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");
```

Availability Zone flexibility for an Amazon EMR cluster

Each AWS Region has multiple, isolated locations known as Availability Zones. When you launch an instance, you can optionally specify an Availability Zone (AZ) in the AWS Region that you use. Availability Zone flexibility is the distribution of instances across multiple AZs. If one instance fails, you can design your application so that an instance in another AZ can handle requests. For more information on Availability Zones, see the Region and zones documentation in the *Amazon EC2 User Guide*.

Instance flexibility is the use of multiple instance types to satisfy capacity requirements. When you express flexibility with instances, you can use aggregate capacity across instance sizes, families, and generations. Greater flexibility improves the chance to find and allocate your required amount of compute capacity when compared with a cluster that uses a single instance type.

Instance and Availability Zone flexibility reduces insufficient capacity errors (ICE) and Spot interruptions when compared to a cluster with a single instance type or AZ. Use the best practices covered here to determine which instances to diversify after you know the initial instance family and size. This approach maximizes availability to Amazon EC2 capacity pools with minimal performance and cost variance.

Being flexible about Availability Zones

We recommend that you configure all Availability Zones for use in your virtual private cloud (VPC) and that you select them for your EMR cluster. Clusters must exist in only one Availability Zone, but with Amazon EMR instance fleets, you can 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 that you specify. When you provision an EMR cluster for multiple subnets, your cluster can access a deeper Amazon EC2 capacity pool when compared to clusters in a single subnet.

If you must prioritize a certain number of Availability Zones for use in your virtual private cloud (VPC) for your EMR cluster, you can leverage the Spot placement score capability with Amazon EC2. With Spot placement scoring, you specify the compute requirements for your Spot Instances, then EC2 returns the top ten AWS Regions or Availability Zones scored on a scale from 1 to 10. A score of 10 indicates that your Spot request is highly likely to succeed; a score of 1 indicates that your Spot request is not likely to succeed. For more information on how to use Spot placement scoring, see Spot placement score in the *Amazon EC2 User Guide*.

Being flexible about instance types

Instance flexibility is the use of multiple instance types to satisfy capacity requirements. Instance flexibility benefits both Amazon EC2 Spot and On-Demand Instance usage. With Spot Instances, instance flexibility lets Amazon EC2 launch instances from deeper capacity pools using real-time capacity data. It also predicts which instances are most available. This offers fewer interruptions and can reduce the overall cost of a workload. With On-Demand Instances, instance flexibility reduces insufficient capacity errors (ICE) when total capacity provisions across a greater number of instance pools.

For **Instance Group** clusters, you can specify up to 50 EC2 instance types. For **Instance Fleets** with allocation strategy, you can specify up to 30 EC2 instance types for each primary, core, and task node group. A broader range of instances improves the benefits of instance flexibility.

Expressing instance flexibility

Consider the following best practices to express instance flexibility for your application.

Topics

- Determine instance family and size
- Include additional instances

Determine instance family and size

Amazon EMR supports several instance types for different use cases. These instance types are listed in the Supported instance types with Amazon EMR documentation. Each instance type belongs to an instance family that describes what application the type is optimized for.

For new workloads, you should benchmark with instance types in the general purpose family, such as `m5` or `c5`. Then, monitor the OS and YARN metrics from Ganglia and Amazon CloudWatch to determine system bottlenecks at peak load. Bottlenecks include CPU, memory, storage, and I/O operations. After you identify the bottlenecks, choose compute optimized, memory optimized, storage optimized, or another appropriate instance family for your instance types. For more details, see the Determine right infrastructure for your Spark workloads page in the Amazon EMR best practices guide on GitHub.

Next, identify the smallest YARN container or Spark executor that your application requires. This is the smallest instance size that fits the container and the minimum instance size for the cluster. Use this metric to determine instances that you can further diversify with. A smaller instance will allow for more instance flexibility.

For maximum instance flexibility, you should leverage as many instances as possible. We recommend that you diversify with instances that have similar hardware specifications. This maximizes access to EC2 capacity pools with minimal cost and performance variance. Diversify across sizes. To do so, prioritize AWS Graviton and previous generations first. As a general rule, try to be flexible across at least 15 instance types for each workload. We recommend that you start with general purpose, compute optimized, or memory optimized instances. These instance types will provide the greatest flexibility.

Include additional instances

For maximum diversity, include additional instance types. Prioritize instance size, Graviton, and generation flexibility first. This allows access to additional EC2 capacity pools with similar cost and performance profiles. If you need further flexibility due to ICE or spot interruptions, consider variant and family flexibility. Each approach has tradeoffs that depend on your use case and requirements.

- **Size flexibility** – First, diversify with instances of different sizes within the same family. Instances within the same family provide the same cost and performance, but can launch a different number of containers on each host. For example, if the minimum executor size that you need is 2vCPU and 8Gb memory, the minimum instance size is `m5.xlarge`. For size flexibility, include

`m5.xlarge`, `m5.2xlarge`, `m5.4xlarge`, `m5.8xlarge`, `m5.12xlarge`, `m5.16xlarge`, and `m5.24xlarge`.

- **Graviton flexibility** – In addition to size, you can diversify with Graviton instances. Graviton instances are powered by AWS Graviton2 processors that deliver the best price performance for cloud workloads in Amazon EC2. For example, with the minimum instance size of `m5.xlarge`, you can include `m6g.xlarge`, `m6g.2xlarge`, `m6g.4xlarge`, `m6g.8xlarge`, and `m6g.16xlarge` for Graviton flexibility.
- **Generation flexibility** – Similar to Graviton and size flexibility, instances in previous generation families share the same hardware specifications. This results in a similar cost and performance profile with an increase in the total accessible Amazon EC2 pool. For generation flexibility, include `m4.xlarge`, `m4.2xlarge`, `m4.10xlarge`, and `m4.16xlarge`.
- **Family and variant flexibility**
  - **Capacity** – To optimize for capacity, we recommend instance flexibility across instance families. Common instances from different instance families have deeper instance pools that can assist with meeting capacity requirements. However, instances from different families will have different vCPU to memory ratios. This results in under-utilization if the expected application container is sized for a different instance. For example, with `m5.xlarge`, include compute-optimized instances such as `c5` or memory-optimized instances such as `r5` for instance family flexibility.
  - **Cost** – To optimize for cost, we recommend instance flexibility across variants. These instances have the same memory and vCPU ratio as the initial instance. The tradeoff with variant flexibility is that these instances have smaller capacity pools which might result in limited additional capacity or higher Spot interruptions. With `m5.xlarge` for example, include AMD-based instances (`m5a`), SSD-based instances (`m5d`) or network-optimized instances (`m5n`) for instance variant flexibility.

Configuring Amazon EMR cluster instance types and best practices for Spot instances

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. The method you should choose depends 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 Use Amazon EMR cluster scaling to adjust for changing workloads.
- **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 Planning and configuring instance fleets for your Amazon EMR cluster.

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 an Amazon EMR cluster as it performs work. 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 primary 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 instance types for use with Amazon EMR.

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 primary 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 primary node (JupyterHub, Hue, etc.), a larger primary 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 primary, 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 primary 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
- Primary node on a Spot Instance
- Core nodes on Spot Instances
- Task nodes on Spot Instances
- Instance configurations for application scenarios

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.

> **Note**
>
> Beginning with Amazon EMR 6.x release series, the YARN node labels feature is disabled by default. The application primary 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'`

- **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 Primary 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`

Primary node on a Spot Instance

The primary node controls and directs the cluster. When it terminates, the cluster ends, so you should only launch the primary 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 primary 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 primary node when you launch the cluster. You can't add or remove primary nodes from a running cluster.

Typically, you would only run the primary 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”.

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 primary 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	Primary node purchasing option	Core nodes purchasing option	Task nodes purchasing option
Long-running clusters and data warehouses	On-Demand	On-Demand or instance-fleet mix	Spot or instance-fleet mix
Cost-driven workloads	Spot	Spot	Spot
Data-critical workloads	On-Demand	On-Demand	Spot or instance-fleet mix
Application testing	Spot	Spot	Spot

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 primary 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 (primary, 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 primary and core instance groups as On-Demand Instances and supplement with one or more task instance groups of Spot Instances. Running the primary 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 (primary, 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*.
- 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 Amazon EMR 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 Amazon EMR cluster primary node using SSH. Amazon EMR collects certain system and applications logs generated by Amazon EMR daemons and other Amazon EMR processes to ensure effective service operations.

> **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.

Console

To archive log files to Amazon S3 with the new console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose **Create cluster**.
3. Under **Cluster logs**, select the **Publish cluster-specific logs to Amazon S3** check box.
4. In the **Amazon S3 location** field, type (or browse to) an Amazon S3 path to store your logs. If you type the name of a folder that doesn't exist in the bucket, Amazon S3 creates it.

When you set this value, 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 terminates the instances hosting the cluster. These logs are useful for troubleshooting purposes. For more information, see View log files.

5. Optionally, select the **Encrypt cluster-specific logs** check box. Then, select an AWS KMS key from the list, enter a key ARN, or create a new key. This option is only available with Amazon EMR version 5.30.0 and later, excluding version 6.0.0. To use this option, add permission to AWS KMS for your EC2 instance profile and Amazon EMR role. For more information, see To encrypt log files stored in Amazon S3 with an AWS KMS customer managed key.
6. Choose any other options that apply to your cluster.
7. To launch your cluster, choose **Create cluster**.

CLI

To archive log files to Amazon S3 with 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.

1. 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-7.10.0 --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
```

2. 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 Amazon EMR service role and EC2 instance profile, enter `aws emr create-default-roles` to create them before typing the `create-cluster` subcommand.

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 Archive log files to Amazon S3. 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**.
- You must also configure the KMS key to allow the `persistentappui.elasticmapreduce.amazonaws.com` and `elasticmapreduce.amazonaws.com` Service Principals to `kms:GenerateDataKey`, `kms:GenerateDataKeyWithoutPlaintext` and `kms:Decrypt`. This allows EMR to read and write logs encrypted with the KMS key to managed storage. The User IAM Role must have permission to use `kms:GenerateDataKey` and `kms:Decrypt`.

```
{
   "Sid": "Allow User Role to use KMS key",
   "Effect": "Allow",
   "Principal": {
        "AWS": "User Role"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
```

```
    ],
     "Resource": "*",
     "Condition": {
         "StringLike": {
             "kms:EncryptionContext:aws:elasticmapreduce:clusterId": "j-*",
            "kms:ViaService": "elasticmapreduce.region.amazonaws.com"
        }
     }
 },
 {
     "Sid": "Allow Persistent APP UI to validate KMS key for write",
     "Effect": "Allow",
     "Principal":{
         "Service": [
             "elasticmapreduce.amazonaws.com"
         ]
      },
      "Action": [
        "kms:GenerateDataKeyWithoutPlaintext"
       ],
      "Resource": "*",
      "Condition": {
         "StringLike": {
             "aws:SourceArn": "arn:aws:elasticmapreduce:region:account:cluster/j-*",
             "kms:EncryptionContext:aws:elasticmapreduce:clusterId": "j-*"
         }
      }
 },
 {
     "Sid": "Allow Persistent APP UI to Write/Read Logs",
     "Effect": "Allow",
     "Principal":{
         "Service": [
             "persistentappui.elasticmapreduce.amazonaws.com",
             "elasticmapreduce.amazonaws.com"
         ]
      },
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": "*",
      "Condition": {
         "StringLike": {
```

```
            "aws:SourceArn": "arn:aws:elasticmapreduce:region:account:cluster/j-*",
            "kms:EncryptionContext:aws:elasticmapreduce:clusterId": "j-*",
            "kms:ViaService": "s3.region.amazonaws.com"
        }
    }
}
```

As a security best practice, we recommend that you add the `kms:EncryptionContext` and `aws:SourceArn` conditions. These conditions help ensure the key is only used by Amazon EMR on EC2 and only used for logs generated from jobs running in a specific cluster.

For more information, see IAM service roles used by Amazon EMR, and Using key policies in the AWS Key Management Service developer guide.

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-7.10.0 \
--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.

Tag and categorize Amazon EMR cluster resources

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 a 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.

Restrictions that apply to tagging resources in Amazon EMR

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 more information about tagging using the Amazon EC2API or command line, see API and CLI overview in the *Amazon EC2 User Guide*.

Tag Amazon EMR 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 an Amazon EMR cluster

You can add tags to a cluster when you create it.

Console

To add tags when you create a cluster with the new console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose **Create cluster**.
3. Under **Tags**, choose **Add new tag**. Specify a tag in the **Key** field. Optionally, specify a tag in the **Value** field.
4. Choose any other options that apply to your cluster.
5. To launch your cluster, choose **Create cluster**.

AWS CLI

To add tags when you create a cluster with the 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.

You can also add tags to an existing cluster.

Console

To add tags to an existing cluster with the new console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.

2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and select the cluster that you want to update.
3. On the **Tags** tab on the cluster details page, select **Manage tags**. Specify a tag in the **Key** field. Optionally, specify a tag in the **Value** field.
4. Select **Save changes**. The **Tags** tab updates with the new number of tags that you have on your cluster. For example, if you now have two tags, the label of your tab is **Tags (2)**.

AWS CLI

To add tags to a running cluster with the AWS CLI

- 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 an Amazon EMR cluster

If you want to see all tags associated with a cluster, you can view them with the console or the AWS CLI.

Console

To view tags on a cluster with the new console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.

2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and select the cluster that you want to update.
3. To view all of your tags, select the **Tags** tab on the cluster details page.

AWS CLI

To view tags on a cluster with 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 an Amazon EMR cluster

If you no longer need a tag, you can remove it from the cluster.

Console

To remove tags on a cluster with the new console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and select the cluster that you want to update.
3. On the **Tags** tab on the cluster details page, select **Manage tags**.

4. Choose **Remove** for each key-value pair that you want to remove.
5. Choose **Save changes**.

AWS CLI

To remove tags on a cluster with 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 on Amazon EMR

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

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

Security and compliance is a responsibility you share with AWS. This shared responsibility model can help relieve your operational burden as AWS operates, manages, and controls the components from the host operating system and virtualization layer down to the physical security of the facilities in which EMR clusters operate. You assume responsibility, management, and updating Amazon EMR clusters, as well as configuring the application software and AWS provided security controls. This differentiation of responsibility is commonly referred to as security *of* the cloud versus security *in* the cloud.

- Security of the cloud – AWS is responsible for protecting the infrastructure that runs AWS services in AWS. 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 – you are also responsible to perform all of the necessary security configuration and management tasks for securing an Amazon EMR cluster. Customers that deploy an Amazon EMR cluster are responsible for management of the application software installed on the instances, and the configuration of the AWS-provided features such as security groups, encryption and access control according to your requirements, applicable laws, and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Amazon EMR. The topics in this chapter show you how to configure Amazon EMR and use other AWS services to meet your security and compliance objectives.

Network and infrastructure security

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. AWS network and infrastructure protection services give you fine-grained protections at both the host and network-level boundaries. Amazon EMR supports AWS services and application features that address your network protection and compliance requirements.

- **Amazon EC2 security groups** act as a virtual firewall for Amazon EMR cluster instances, limiting inbound and outbound network traffic. For more information, see Control network traffic with security groups.
- **Amazon EMR block public access (BPA)** prevents you from launching a cluster in a public subnet if the cluster has a security configuration that allows inbound traffic from public IP addresses on a port. For more information, see Using Amazon EMR block public access.
- **Secure Shell (SSH)** helps provide a secure way for users to connect to the command line on cluster instances. You can also use SSH to view web interfaces that applications host on the master node of a cluster. For more information, see Use an EC2 key pair for SSH credentials and Connect to a cluster.

Updates to the default Amazon Linux AMI for Amazon EMR

> **Important**
>
> EMR clusters that run Amazon Linux or Amazon Linux 2 Amazon Machine Images (AMIs) 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 that run 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 release becomes available, EMR cluster instances that run 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. Note that your networking configuration must allow for HTTP and HTTPS egress to Linux repositories in Amazon S3, otherwise security updates will not succeed.

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 identities include users, groups, and roles. An IAM role is similar to an IAM user, but is not associated with a specific person, and is intended to be assumable by any user who needs permissions. For more information, see AWS Identity and Access Management for Amazon EMR. Amazon EMR uses multiple IAM roles to help you implement access controls for Amazon EMR clusters. IAM is an AWS service that you can use with no additional charge.

- **IAM role for Amazon EMR (EMR role)** – controls how Amazon EMR service is able to access other AWS services on your behalf, such as provisioning Amazon EC2 instances when the Amazon EMR cluster launches. For more information, see Configure IAM service roles for Amazon EMR permissions to AWS services and resources.
- **IAM role for cluster EC2 instances (EC2 instance profile)** – a role that is assigned to every EC2 instance in the Amazon EMR cluster when the instance launches. Application processes that run on the cluster use this role to interact with other AWS services, such as Amazon S3. For more information, see IAM role for cluster's EC2 instances.
- **IAM role for applications (runtime role)** – an IAM role that you can specify when you submit a job or query to an Amazon EMR cluster. The job or query that you submit to your Amazon EMR cluster uses the runtime role to access AWS resources, such as objects in Amazon S3. You can specify runtime roles with Amazon EMR for Spark and Hive jobs. Bu using runtime roles, you can isolate jobs running on the same cluster by using different IAM roles. For more information, see Using IAM role as runtime role with Amazon EMR.

Workforce identities refer to users who build or operate workloads in AWS. Amazon EMR provides support for workforce identities with the following:

- **AWS IAM identity center (Idc)** is the recommended AWS service for managing user access to AWS resources. It is a single place where you can assign your workforce identities, consistent access to multiple AWS accounts and applications. Amazon EMR supports workforce identities through trusted identity propagation. With trusted identity propagation capability, a user can sign in to the application and that application can pass the identity of the user to other AWS services for authorizing access to data or resources. For more information see, Enabling support for AWS IAM identity center with Amazon EMR.

  Lightweight Directory Access Protocol (LDAP) is an open, vendor-neutral, industry standard application protocol for accessing and maintaining information about users, systems, services, and applications over the network. LDAP is commonly used for user authentication against corporate identity servers such as Active Directory (AD) and OpenLDAP. By enabling LDAP with

EMR clusters, you allow you users use their existing credentials to authenticate and access clusters. For more information see, enabling support for LDAP with Amazon EMR.

Kerberos is a network authentication protocol designed to provide strong authentication for client/server applications by using secret-key cryptography. When you use Kerberos, Amazon EMR configures Kerberos for the applications, components, and subsystems that it installs on the cluster so that they are authenticated with each other. To access a cluster with Kerberos configured, a kerberos principal must be present in the Kerberos Domain Controller (KDC). For more information, see enabling support for Kerberos with Amazon EMR.

Single-tenant and multi-tenant clusters

A cluster is by default configured for a single tenancy with the EC2 Instance profile as the IAM identity. In a single-tenant cluster, every job has full and complete access to the cluster and access to all AWS services and resources is done on the basis of the EC2 instance profile. In a multi-tenant cluster, tenants are isolated from each other and the tenants don't have full and complete access to the clusters and EC2 Instances of the cluster. The identity on multi-tenant clusters is either the runtime roles or the workforce identifies. In a multi-tenant cluster, you can also enable support for fine-grained access control (FGAC) via AWS Lake Formation or Apache Ranger. A cluster that has runtime roles or FGAC enabled, access to the EC2 Instance profile is also disable via iptables.

> **Important**
>
> Any users who have access to a single-tenant cluster can install any software on the Linux operating system (OS), change or remove software components installed by Amazon EMR and impact the EC2 Instances that are part of the cluster. If you want to ensure that users can't install or change configurations of an Amazon EMR cluster, we recommend that you enable multi-tenancy for the cluster. You can enable multi-tenancy on a cluster by enabling support for runtime role, AWS IAM identity center, Kerberos, or LDAP.

Data protection

With AWS, you control your data by using AWS services and tools to determine how the data is secured and who has access to it. Services such as AWS Identity and Access Management (IAM) let you securely manage access to AWS services and resources. AWS CloudTrail enables detection and auditing. Amazon EMR makes it easy for you to encrypt data at rest in Amazon S3 by using keys

either managed by AWS or fully managed by you. Amazon EMR also support enabling encryption for data in transit. For more information, see encrypt data at rest and in transit.

Data Access Control

With data access control, you can control what data an IAM identity or a workforce identity can access. Amazon EMR supports the following access controls:

- **IAM identity-based policies** – manage permissions for IAM roles that you use with Amazon EMR. IAM 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.
- **AWS Lake Formation** centralizes permissions management of your data and makes it easier to share across your organization and externally. You can use Lake Formation to enable fine-grained, column-level access to databases and tables in the AWS Glue Data Catalog. For more information, see Using AWS Lake Formation with Amazon EMR.
- **Amazon S3 access grants** map identities map identities in directories such as Active Directory, or AWS Identity and Access Management (IAM) principals, to datasets in S3. Additionally, S3 access grants log end-user identity and the application used to access S3 data in AWS CloudTrail. For more information, see Using Amazon S3 access grants with Amazon EMR.
- **Apache Ranger** is a framework to enable, monitor and manage comprehensive data security across the Hadoop platform. Amazon EMR supports Apache Ranger based fine-grained access control for Apache Hive Metastore and Amazon S3. For more information see Integrate Apache Ranger with Amazon EMR.

Use security configurations to set up Amazon EMR cluster security

You can use Amazon EMR security configurations to configure data encryption, Kerberos authentication, and Amazon S3 authorization for EMRFS on your clusters. First, you create a security configuration. Then, the security configuration is available to use and re-use when you create clusters.

You can use the AWS Management 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 with the Amazon EMR console or with the AWS CLI
- Specify a security configuration for an Amazon EMR cluster

Create a security configuration with the Amazon EMR console or with the AWS CLI

This topic covers general procedures to create a security configuration with the Amazon 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 with Amazon EMR
- Use Kerberos for authentication with Amazon EMR
- Configure IAM roles for EMRFS requests to Amazon S3

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 and Providing certificates for encrypting data in transit with Amazon EMR encryption.

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 with Amazon EMR.

> **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.

- **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.

    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 Amazon 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.

  - **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 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 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 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": "amzn-s3-demo-bucket1",
                "EncryptionMode": "SSE-S3"
              },
              {
                "BucketName": "amzn-s3-demo-bucket2",
                "EncryptionMode": "CSE-KMS",
                "AwsKmsKey": "arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012"
              },
              {
                "BucketName": "amzn-s3-demo-bucket3",
                "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"
            }
        }
     }
}'
```

The example below illustrates the following scenario:

SSE-EMR-WAL is used for EMR WAL encryption

```
aws emr create-security-configuration --name "MySecConfig" \
    --security-configuration '{
        "EncryptionConfiguration": {
            "EMRWALEncryptionConfiguration":{ },
            "EnableInTransitEncryption":false, "EnableAtRestEncryption":false
        }
    }'
```

`EnableInTransitEncryption` and `EnableAtRestEncryption` still could be true, if want to enable related encryption.

The example below illustrates the following scenario:

- SSE-KMS-WAL is used for EMR WAL encryption
- Server side encryption uses AWS Key Management Service as the key provider

```
aws emr create-security-configuration --name "MySecConfig" \
    --security-configuration '{
        "EncryptionConfiguration": {
            "EMRWALEncryptionConfiguration":{
                "AwsKmsKey":"arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012"
                },
            "EnableInTransitEncryption":false, "EnableAtRestEncryption":false
        }
    }'
```

`EnableInTransitEncryption` and `EnableAtRestEncryption` still could be true, if want to enable related encryption.

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

Parameter	Description
	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	
`"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 Amazon 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.

Parameter	Description
"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/12345678-1234-1234-1234-123456789012).
"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 .

Parameter	Description
"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.
"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/12345678-1234-1234-1234-456789012123).
"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* .
EMR WAL encryption parameters	
"EMRWALEncryptionConfiguration"	Specifies the value for EMR WAL encryption.
"AwsKmsKey"	Specifies the CMK Key Id Arn.

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 for authentication with Amazon EMR. 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 primary 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 configura tion 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.

Parameter		Description
		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 primary 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 configura tion. Additional configuration in the other Kerberos realm is required. For more information, see Tutorial: Configure a cross-realm trust with an Active Directory domain.
Cross-rea lm 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`*).

Parameter		Description
	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`*).
	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.

Parameter			Description
	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.

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 primary 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

Parameter	Description
	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 primary 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 primary node command line to renew after a ticket expires.

Parameter	Description
`"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.
`"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.

Parameter				Description
			`"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 primary 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.
			"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`*).

Parameter	Description
"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.
`}`	
`}`	
`}`	

Parameter	Description
}	

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.

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_demo_s3_buckets",
        "IdentifierType": "Prefix",
        "Identifiers": [ "s3://amzn-s3-demo-bucket1/","s3://amzn-s3-demo-bucket2/" ]
      },{
        "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://amzn-s3-demo-bucket/` matches `s3://amzn-s3-demo-bucket/mydir` and

Parameter	Description
	`s3://amzn-s3-demo-bucket/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*.

> **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. Using a custom security configuration is optional.

```
{
  "InstanceMetadataServiceConfiguration" : {
      "MinimumInstanceMetadataServiceVersion": integer,
      "HttpPutResponseHopLimit": integer
   }
}
```

Parameter	Description
`"InstanceMetadataServiceConfiguration":`	If you don't specify IMDS within a security configuration and use an Amazon EMR release that requires IMDSv1, Amazon EMR defaults to using IMDSv1 as the minimum instance metadata service version. If you want to use your own configuration, both of the following parameters are required.
`"MinimumInstanceMetadataServiceVersion":`	Required. Specify 1 or 2. A value of 1 allows IMDSv1 and IMDSv2. A value of 2 allows only IMDSv2.
`"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*.
   - **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*.

See Configure instance details and Configure the instance metadata service in the *Amazon EC2 User Guide*.

Specify a security configuration for an Amazon EMR 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.

Console

To specify a security configuration with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose **Create cluster**.
3. Under **Security configuration and permissions**, find the **Security configuration** field. Select the dropdown menu or choose **Browse** to select the name of a security configuration that you created previously. Alternatively, choose **Create security configuration** to create a configuration that you can use for your cluster.
4. Choose any other options that apply to your cluster.

5. To launch your cluster, choose **Create cluster**.

CLI

To specify a security configuration with the AWS CLI

- Use `aws emr create-cluster` to optionally apply a security configuration with `--security-configuration` *`MySecConfig`*, where *`MySecConfig`* is the name of the security configuration, as shown in the following example. The `--release-label` you specify 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 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 with Amazon EMR

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

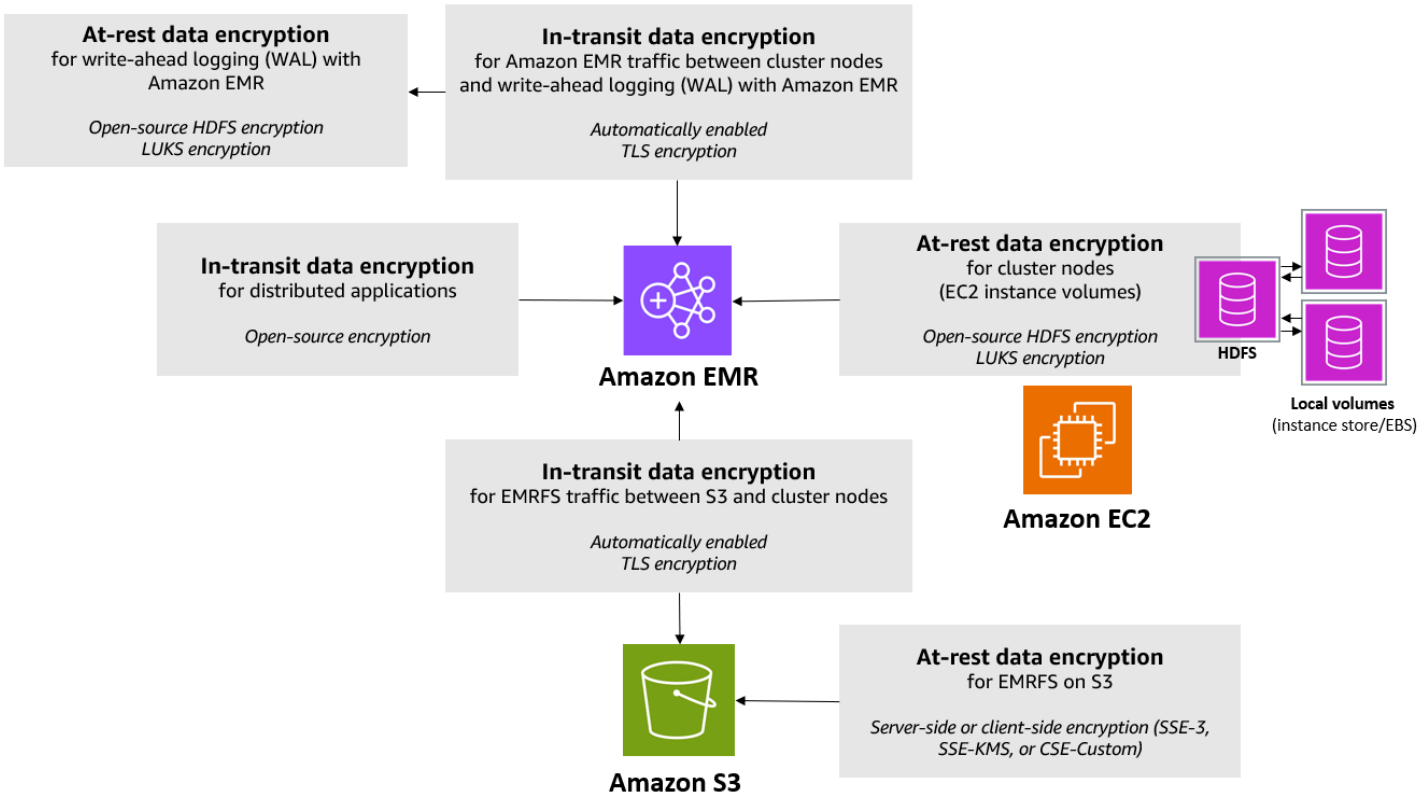
- Encryption options for Amazon EMR
- Encryption at rest using a customer KMS key for the EMR WAL service
- Create keys and certificates for data encryption with Amazon EMR
- Understanding in-transit encryption

Encryption options for Amazon EMR

With Amazon EMR releases 4.8.0 and higher, 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 with the Amazon EMR console or with the AWS CLI.

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.

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 the Amazon 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 more 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

All Amazon S3 buckets have encryption configured by default, and all new objects that are uploaded to an S3 bucket are automatically encrypted at rest, 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.

Encryption at rest for data in Amazon EMR WAL

When you set up server-side encryption (SSE) for write-ahead logging (WAL), Amazon EMR encrypts data at rest. You can choose from two different key management systems when you specify SSE in Amazon EMR:

SSE-EMR-WAL

Amazon EMR manages keys for you. By default, Amazon EMR encrypts the data that you stored in Amazon EMR WAL with SSE-EMR-WAL.

SSE-KMS-WAL

You use an AWS KMS key to set up policies that apply to Amazon EMR WAL. For more information about configuring encryption at rest for EMR WAL using a customer KMS key, see Encryption at rest using a customer KMS key for the EMR WAL service.

> **Note**
>
> You can't use your own key with SSE when you enable WAL with Amazon EMR. For more information, see Write-ahead logs (WAL) for Amazon EMR.

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. 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 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*. 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 might vary by Amazon EMR release. To enable in-transit encryption, use the section called "Create a security configuration with the Amazon EMR console or with the AWS CLI" in Amazon EMR. For EMR clusters with in-transit encryption enabled, Amazon EMR automatically configures the open-source application configurations to enable in-transit encryption. For advanced use cases, you can configure open-source application configurations directly to override the default behavior in Amazon EMR. For more information, see in-transit encryption support matrix and Configure applications.

See the following to learn more specific details about open-source applications relevant to in-transit encryption:

- When you enable in-transit encryption with a security configuration, Amazon EMR enables in-transit encryption for all open-source application endpoints that support in-transit encryption. Support for in-transit encryption for different application endpoints varies by the Amazon EMR release version. For more information, see the in-transit encryption support matrix.
- You can override open-source configurations, which lets you do the following:
  - Disable TLS hostname verification if your user-provided TLS certificates doesn't meet requirements
  - Disable in-transit encryption for certain endpoints based on your performance and compatibility requirements
  - Control which TLS versions and cipher suites to use.

  You can find more details about the application-specific configurations in the in-transit encryption support matrix
- Aside from enabling in-transit encryption with a security configuration, some communication channels also require additional security configurations for you to enable in-transit encryption. For example, some open-source application endpoints use Simple Authentication and Security

Layer (SASL) for in-transit encryption, which requires that Kerberos authentication is enabled in the security configuration of the EMR cluster. To learn more about these endpoints, see the in-transit encryption support matrix.

- We recommend that you use software that support TLS v1.2 or higher. Amazon EMR on EC2 ships the default Corretto JDK distribution, which determines which TLS versions, cipher suites, and key sizes are allowed by the open-source networks that run on Java. At this time, most open-source frameworks enforce TLS v1.2 or higher for Amazon EMR 7.0.0 and higher releases. This is because most open-source frameworks run on Java 17 for Amazon EMR 7.0.0 and higher. Older Amazon EMR release versions might support TLS v1.0 and v1.1 because they consume older Java versions, but Corretto JDK might change which TLS versions that Java supports, which might impact existing Amazon EMR releases.

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.

Encryption at rest using a customer KMS key for the EMR WAL service

EMR write-ahead logs (WAL) provides customer KMS key encryption-at-rest support. The following details at a high level how Amazon EMR WAL is integrated with AWS KMS:

The EMR write-ahead logs (WAL) interact with AWS during the following operations: `CreateWAL`, `AppendEdit`, `ArchiveWALCheckPoint`, `CompleteWALFlush`, `DeleteWAL`, `GetCurrentWALTime`, `ReplayEdits`, `TrimWAL` via the `EMR_EC2_DefaultRole` by default When any the previous operations listed are invoked, the EMR WAL makes `Decrypt` and `GenerateDataKey` against the KMS key.

Considerations

Consider the following when using AWS KMS based encryption for EMR WAL:

- The encryption configuration can't be changed after an EMR WAL is created.
- When you use KMS encryption with your own KMS key, the key must exist in the same region as your Amazon EMR cluster.
- You are responsible to maintain all required IAM permissions and it's recommended to not revoke the needed permissions during the life of the WAL. Otherwise, it will cause unexpected failure scenarios, such as the inability to delete EMR WAL, as the associated encryption key doesn't exist.

- There is a cost associated with using AWS KMS keys. For more information, see AWS Key Management Service pricing.

Required IAM permissions

To use your customer KMS key to encrypt EMR WAL at rest, you need to make sure you set proper permission to the EMR WAL client role and the EMR WAL service principal `emrwal.amazonaws.com`.

Permissions for the EMR WAL client role

Below is the IAM policy needed for the EMR WAL client role:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey*"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowKMSDecrypt"
    }
  ]
}
```

The EMR WAL client on EMR cluster will use `EMR_EC2_DefaultRole` by default. If you use a different role for the instance profile in the EMR cluster, make sure that each role has appropriate permissions.

For more information about managing the role policy, refer to Adding and removing IAM identity permissions.

Permissions for the KMS key policy

You need to give the EMR WAL client role and EMR WAL service `Decrypt` and `GenerateDataKey*` permission in your KMS policy. For more about the key policy management, refer to KMS key policy.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey*"
      ],
      "Resource": [
        "arn:aws:kms:*:123456789012:key/*"
      ],
      "Sid": "AllowKMSDecrypt"
    }
  ]
}
```

The role specified in the snippet can change if you change the default role.

Monitoring Amazon EMR WAL interaction with AWS KMS

Amazon EMR WAL encryption context

An encryption context is a set of key–value pairs that contains arbitrary non-secret data. When you include an encryption context in a request to encrypt data, AWS KMS cryptographically binds the encryption context to the encrypted data. To decrypt the data, you must pass in the same encryption context.

In its GenerateDataKey and Decrypt requests to AWS KMS, Amazon EMR WAL uses an encryption context with one name–value pairs that identify the EMR WAL name.

```
"encryptionContext": {
```

```
    "aws:emrwal:walname": "111222333444555-testworkspace-emrwalclustertest-
emrwaltestwalname"
}
```

You can use the encryption context to identify these cryptographic operation in audit records and logs, such as AWS CloudTrail and Amazon CloudWatch Logs, and as a condition for authorization in policies and grants.

Create keys and certificates for data encryption with Amazon EMR

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

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 Amazon 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 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 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 following procedure describes how to use the AWS Management Console to add the default Amazon EMR service role `EMR_DefaultRole` as a *key user*. 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 Amazon 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** section, select the appropriate role. The name of the default service role for Amazon EMR 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. The requirements for custom key provider depend on whether you use local disk encryption and Amazon S3 encryption, as well as the Amazon EMR release version.

Depending on the type of encryption you use when creating a custom key provider, the application must also implement different EncryptionMaterialsProvider interfaces. Both interfaces are available in the AWS SDK for Java version 1.11.0 and later.

- To implement Amazon S3 encryption, use the com.amazonaws.services.s3.model.EncryptionMaterialsProvider interface.
- To implement local disk encryption, use the com.amazonaws.services.elasticmapreduce.spi.security.EncryptionMaterialsProvider interface.

You can use any strategy to provide encryption materials for the implementation. For example, you might choose to provide static encryption materials or integrate with a more complex key management system.

If you're using Amazon S3 encryption, you must use the encryption algorithms **AES/GCM/NoPadding** for custom encryption materials.

If you're using local disk encryption, the encryption algorithm to use for custom encryption materials varies by EMR release. For Amazon EMR 7.0.0 and lower, you must use **AES/GCM/NoPadding**. For Amazon EMR 7.1.0 and higher, you must use **AES**.

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.

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	We recommend that you provide a certificate that isn't signed by the the Java default trusted root certification authority (CA) or an intermediate CA that can link to the Java default trusted root CA. We don't reocmmend that you use public CAs when you use wildcard certificates or when you disable hostname verification.

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 for the domain in the common name, you must change the value of `hadoop.ssl.hostname.verifier` to `ALLOW_ALL`. To do so in Amazon EMR releases 7.3.0 and higher, add the `core-site` classification when you submit configurations to a cluster. In releases lower than 7.3.0, add the configuration `"hadoop.ssl.hostname.verifier": "ALLOW_ALL"` directly into the `core-site.xml` file. This change is required because the default hostname verifier requires a hostname without the wildcard because all hosts in the cluster use it. For more information about EMR cluster configuration within an Amazon VPC, see Configure networking in a VPC for Amazon EMR.

The following example demonstrates how to use OpenSSL to generate a self-signed X.509 certificate with a 2048-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:2048 -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
```

Understanding in-transit encryption

You can configure an EMR cluster to run open-source frameworks such as Apache Spark, Apache Hive, and Presto. each of these open-source frameworks has a set of processes running on the EC2 instances of a cluster. Each of these processes can host network endpoints for network communication.

If in-transit encryption is enabled on an EMR cluster, different network endpoints use different encryption mechanisms. See the following sections to learn more about the specific open-source framework network endpoints supported with in-transit encryption, the related encryption mechanisms, and which Amazon EMR release added the support. Each open-source application might also have different best practices and open-source framework configurations that you can change.

For the most in-transit encryption coverage, we recommend that you enable both in-transit encryption and Kerberos. If you only enable in-transit encryption, then in-transit encryption will be available only for the network endpoints that support TLS. Kerberos is necessary because some open-source framework network endpoints use Simple Authentication and Security Layer (SASL) for in-transit encryption.

Note that any open-source frameworks not supported in Amazon EMR 7.x.x releases are not included.

Spark

When you enable in-transit encryption in security configurations, `spark.authenticate` is automatically set to `true` and uses AES-based encryption for RPC connections.

Starting with Amazon EMR 7.3.0, if you use in-transit encryption and Kerberos authentication, you can't use Spark applications that depend on the Hive metastore. Hive 3 fixes this issue in HIVE-16340. HIVE-44114 fully resolves this issue when open-source Spark can upgrade to Hive 3. In the meantime, you can set `hive.metastore.use.SSL` to `false` to work around this issue. For more information, see Configure applications.

For more information, see Spark security in the Apache Spark documentation.

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release
Spark History Server	spark.ssl.history.port	18480	TLS	emr-5.3.0+, emr-6.0.0+, emr-7.0.0+
Spark UI	spark.ui.port	4440	TLS	emr-5.3.0+, emr-6.0.0+, emr-7.0.0+
Spark Driver	spark.driver.port	Dynamic	Spark AES-based encryption	emr-4.8.0+, emr-5.0.0+, emr-6.0.0+, emr-7.0.0+
Spark Executor	Executor Port (no named config)	Dynamic	Spark AES-based encryption	emr-4.8.0+, emr-5.0.0+, emr-6.0.0+, emr-7.0.0+
YARN NodeManager	spark.shuffle.service.port[1]	7337	Spark AES-based encryption	emr-4.8.0+, emr-5.0.0+, emr-6.0.0+, emr-7.0.0+

[1] `spark.shuffle.service.port` is hosted on YARN NodeManager but is only used by Apache Spark.

Hadoop YARN

Secure Hadoop RPC is set to `privacy` and uses SASL-based in-transit encryption. This requires that Kerberos authentication is enabled in the security configuration. If you don't want in-transit encryption for Hadoop RPC, configure `hadoop.rpc.protection = authentication`. We recommend that you use the default configuration for maximum security.

If your TLS certificates can't meet TLS hostname verification requirements, you can configure `hadoop.ssl.hostname.verifier = ALLOW_ALL`. We recommend that you use the default configuration of `hadoop.ssl.hostname.verifier = DEFAULT`, which enforces TLS hostname verification.

To disable HTTPS for the YARN web application endpoints, configure `yarn.http.policy = HTTP_ONLY`. This makes it so that traffic to these endpoints stays unencrypted. We recommend that you use the default configuration for maximum security.

For more information, see Hadoop in secure mode in the Apache Hadoop documentation.

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release
ResourceManager	yarn.resourcemanager.webapp.address	8088	TLS	emr-7.3.0+
ResourceManager	yarn.resourcemanager.resource-tracker.address	8025	SASL + Kerberos	emr-4.8.0+, emr-5.0.0+, emr-6.0.0+, emr-7.0.0+
ResourceManager	yarn.resourcemanager.scheduler.address	8030	SASL + Kerberos	emr-4.8.0+, emr-5.0.0+, emr-6.0.0+, emr-7.0.0+
ResourceManager	yarn.resourcemanager.address	8032	SASL + Kerberos	emr-4.8.0+, emr-5.0.0+, emr-6.0.0+, emr-7.0.0+

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release
ResourceManager	yarn.resourcemanager.admin.address	8033	SASL + Kerberos	emr-4.8.0+, emr-5.0.0+, emr-6.0.0+, emr-7.0.0+
TimelineServer	yarn.timeline-service.address	10200	SASL + Kerberos	emr-4.8.0+, emr-5.0.0+, emr-6.0.0+, emr-7.0.0+
TimelineServer	yarn.timeline-service.webapp.address	8188	TLS	emr-7.3.0+
WebApplicationProxy	yarn.web-proxy.address	20888	SASL + Kerberos	emr-4.8.0+, emr-5.0.0+, emr-6.0.0+, emr-7.0.0+
NodeManager	yarn.nodemanager.address	8041	SASL + Kerberos	emr-4.8.0+, emr-5.0.0+, emr-6.0.0+, emr-7.0.0+
NodeManager	yarn.nodemanager.localizer.address	8040	SASL + Kerberos	emr-4.8.0+, emr-5.0.0+, emr-6.0.0+, emr-7.0.0+

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release
NodeManager	yarn.node manager.w ebapp.address	8044	TLS	emr-7.3.0+
NodeManager	mapreduce .shuffle.port[1]	13562	TLS	emr-4.8.0+, emr-5.0.0+, emr-6.0.0+, emr-7.0.0+
NodeManager	spark.shu ffle.service.port[2]	7337	Spark AES-based encryption	emr-4.8.0+, emr-5.0.0+, emr-6.0.0+, emr-7.0.0+

[1] `mapreduce.shuffle.port` is hosted on YARN NodeManager but is only used by Hadoop MapReduce.

[2] `spark.shuffle.service.port` is hosted on YARN NodeManager but is only used by Apache Spark.

Known issue

The `yarn.log.server.url` configuration in is currently using HTTP with port 19888, which prevents access to application logs from the Resource Manager UI. **Affects Version:** EMR - 7.3.0 to EMR - 7.8.0.

Use the following workaround:

1. Modify the `yarn.log.server.url` configuration in `yarn-site.xml` to use the `HTTPS` protocol and port number `19890`.
2. Restart YARN Resource Manager: `sudo systemctl restart hadoop-yarn-resourcemanager.service`.

Hadoop HDFS

The Hadoop name node, data node, and journal node all support TLS by default if in-transit encryption is enabled in EMR clusters.

Secure Hadoop RPC is set to to `privacy` and uses SASL-based in-transit encryption. This requires that Kerberos authentication is enabled in the security configuration.

We recommend that you don't change the default ports used for HTTPS endpoints.

Data encryption on HDFS block transfer uses AES 256 and requires that at-rest encryption is enabled in the security configuration.

For more information, see Hadoop in secure mode in the Apache Hadoop documentation.

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release
Namenode	dfs.namenode.https-address	9871	TLS	emr-4.8.0+, emr-5.0.0+, emr-6.0.0+, emr-7.0.0+
Namenode	dfs.namenode.rpc-address	8020	SASL + Kerberos	emr-4.8.0+, emr-5.0.0+, emr-6.0.0+, emr-7.0.0+
Datanode	dfs.datanode.https.address	9865	TLS	emr-4.8.0+, emr-5.0.0+, emr-6.0.0+, emr-7.0.0+
Datanode	dfs.datanode.address	9866	SASL + Kerberos	emr-4.8.0+, emr-5.0.0+,

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release
				emr-6.0.0+, emr-7.0.0+
Journal Node	dfs.journalnode.https-address	8481	TLS	emr-4.8.0+, emr-5.0.0+, emr-6.0.0+, emr-7.0.0+
Journal Node	dfs.journalnode.rpc-address	8485	SASL + Kerberos	emr-4.8.0+, emr-5.0.0+, emr-6.0.0+, emr-7.0.0+
DFSZKFailoverController	dfs.ha.zkfc.port	8019	None	TLS for ZKFC is only supported in Hadoop 3.4.0. See HADOOP-18919 for more information. Amazon EMR release 7.1.0 is currently on Hadoop 3.3.6. Higher Amazon EMR releases are on Hadoop 3.4.0 in the future

Hadoop MapReduce

Hadoop MapReduce, job history server, and MapReduce shuffle all support TLS by default when in-transit encryption is enabled in EMR clusters.

Hadoop MapReduce encrypted shuffle uses TLS.

We recommend that you don't change the default ports for HTTPS endpoints.

For more information, see Hadoop in secure mode in the Apache Hadoop documentation.

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release
JobHistoryServer	mapreduce.jobhistory.webapp.https.address	19890	TLS	emr-7.3.0+
YARN NodeManager	mapreduce.shuffle.port[1]	13562	TLS	emr-4.8.0+, emr-5.0.0+, emr-6.0.0+, emr-7.0.0+

[1] `mapreduce.shuffle.port` is hosted on YARN NodeManager but is only used by Hadoop MapReduce.

Presto

In Amazon EMR releases 5.6.0 and higher, internal communication between the Presto coordinator and workers uses TLS Amazon EMR sets up all the required configurations to enable secure internal communication in Presto.

If the connector uses the Hive metastore as the metadata store, communication between the communicator and the Hive metastore is also encrypted with TLS.

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release
Presto Coordinator	http-server.https.port	8446	TLS	emr-5.6.0+, emr-6.0.0+, emr-7.0.0+
Presto Worker	http-server.https.port	8446	TLS	emr-5.6.0+, emr-6.0.0+, emr-7.0.0+

Trino

In Amazon EMR releases 6.1.0 and higher, internal communication between the Presto coordinator and workers uses TLS Amazon EMR sets up all the required configurations to enable secure internal communication in Trino.

If the connector uses the Hive metastore as the metadata store, communication between the communicator and the Hive metastore is also encrypted with TLS.

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release
Trino Coordinator	http-server.https.port	8446	TLS	emr-6.1.0+, emr-7.0.0+
Trino Worker	http-server.https.port	8446	TLS	emr-6.1.0+, emr-7.0.0+

Hive and Tez

By default, Hive server 2, Hive metastore server, Hive LLAP Daemon web UI, and Hive LLAP shuffle all support TLS when in-transit encryption is enabled in the EMR clusters. For more information about the Hive configurations, see Configuration properties.

Tez UI that's hosted on the Tomcat server is also HTTPS-enabled when in-transit encryption is enable in the EMR cluster. However, HTTPS is disabled for the Tez AM web UI service so AM users don't have access to the keystore file for the opening SSL listener. You can also enable this behavior with the Boolean configurations `tez.am.tez-ui.webservice.enable.ssl` and `tez.am.tez-ui.webservice.enable.client.auth`.

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release
HiveServer2	hive.server2.thrift.port	10000	TLS	emr-6.9.0+, emr-7.0.0+
HiveServer2	hive.server2.thrift.http.port	10001	TLS	emr-6.9.0+, emr-7.0.0+
HiveServer2	hive.server2.webui.port	10002	TLS	emr-7.3.0+
HiveMetastoreServer	hive.metastore.port	9083	TLS	emr-7.3.0+
LLAP Daemon	hive.llap.daemon.yarn.shuffle.port	15551	TLS	emr-7.3.0+
LLAP Daemon	hive.llap.daemon.web.port	15002	TLS	emr-7.3.0+

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release
LLAP Daemon	hive.llap.daemon.output.service.port	15003	None	Hive doesn't support in-transit encryption for this endpoint
LLAP Daemon	hive.llap.management.rpc.port	15004	None	Hive doesn't support in-transit encryption for this endpoint
LLAP Daemon	hive.llap.plugin.rpc.port	Dynamic	None	Hive doesn't support in-transit encryption for this endpoint
LLAP Daemon	hive.llap.daemon.rpc.port	Dynamic	None	Hive doesn't support in-transit encryption for this endpoint
WebHCat	templeton.port	50111	TLS	emr-7.3.0+
Tez Application Master	tez.am.client.am.port-range tez.am.task.am.port-range	Dynamic	None	Tez doesn't support in-transit encryption for this endpoint

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release
Tez Application Master	tez.am.tez-ui.webservice.port-range	Dynamic	None	Disabled by default. Can be enabled using Tez configurations in emr-7.3.0+
Tez Task	N/A - not configurable	Dynamic	None	Tez doesn't support in-transit encryption for this endpoint
Tez UI	Configurable via Tomcat server on which Tez UI is hosted	8080	TLS	emr-7.3.0+

Flink

Apache Flink REST endpoints and internal communication between flink processes support TLS by default when you enable in-transit encryption in EMR clusters.

`security.ssl.internal.enabled` is set to `true` and uses in-transit encryption for internal communication between the Flink processes. If you don't want in-transit encryption for internal communication, disable that configuration. We recommend you use the default configuration for maximum security.

Amazon EMR sets `security.ssl.rest.enabled` to `true` and uses in-transit encryption for the REST endpoints. Additionally, Amazon EMR also sets `historyserver.web.ssl.enabled` to true to use TLS communication with the Flink history server. If you don't want in-transit encryption for the REST points, disable these configurations. We recommend you use the default configuration for maximum security.

Amazon EMR uses `security.ssl.algorithms`. to specify the list of ciphers that use AES-based encryption. Override this configuration to use the ciphers you want.

For more information, see SSL Setup in the Flink documentation.

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release
Flink History Server	historyserver.web.port	8082	TLS	emr-7.3.0+
Job Manager Rest Server	rest.bind-port rest.port	Dynamic	TLS	emr-7.3.0+

HBase

Amazon EMR sets Secure Hadoop RPC to `privacy`. HMaster and RegionServer use SASL-based in-transit encryption. This requires that Kerberos authentication is enabled in the security configuration.

Amazon EMR sets `hbase.ssl.enabled` to true and uses TLS for UI endpoints. If you don't want to use TLS for UI endpoints, disable this configuration. We recommend that you use the default configuration for maximum security.

Amazon EMR sets `hbase.rest.ssl.enabled` and `hbase.thrift.ssl.enabled` and uses TLS for the REST and Thirft server endpoints, respectively. If you don't want to use TLS for these endpoints, disable this configuration. We recommend that you use the default configuration for maximum security.

Starting with EMR 7.6.0, TLS is supported on HMaster and RegionServer endpoints. Amazon EMR also sets `hbase.server.netty.tls.enabled` and `hbase.client.netty.tls.enabled`. If you don’t want to use TLS for these endpoints, disable this configuration. We recommend that you use the default configuration, which provides encryption and thus higher security. To learn more, see Transport Level Security (TLS) in HBase RPC communication in the *Apache HBase Reference Guide*.

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release
HMaster	HMaster	16000	SASL + Kerberos TLS	SASL + Kerberos in emr-4.8.0+, emr-5.0.0+, emr-6.0.0+, and emr-7.0.0+ TLS in emr-7.6.0+
HMaster	HMaster UI	16010	TLS	emr-7.3.0+
RegionServer	RegionServer	16020	SASL + Kerberos TLS	SASL + Kerberos in emr-4.8.0+, emr-5.0.0+, emr-6.0.0+, and emr-7.0.0+ TLS in emr-7.6.0+
RegionServer	RegionServer Info	16030	TLS	emr-7.3.0+
HBase Rest Server	Rest Server	8070	TLS	emr-7.3.0+
HBase Rest Server	Rest UI	8085	TLS	emr-7.3.0+
	Thrift Server	9090	TLS	emr-7.3.0+

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release
Hbase Thrift Server				
Hbase Thrift Server	Thrift Server UI	9095	TLS	emr-7.3.0+

Phoenix

If you enabled in-transit encryption in your EMR cluster, Phoenix Query Serversupports the TLS property `phoenix.queryserver.tls.enabled`, which is set to `true` by default.

To learn more, see Configurations relating to HTTPS in the Phoenix Query Server documentation.

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release
Query Server	phoenix.queryserver.http.port	8765	TLS	emr-7.3.0+

Oozie

OOZIE-3673 is available on Amazon EMR if you run Oozie on Amazon EMR 7.3.0 and higher. If you need to configure custom SSL or TLS protocols when you run an email action, you can set the property `oozie.email.smtp.ssl.protocols` in the `oozie-site.xml` file. By default, if you enabled in-transit encryption, Amazon EMR uses the TLS v1.3 protocol.

OOZIE-3677 and OOZIE-3674 are also available on Amazon EMR if you run Oozie on Amazon EMR 7.3.0 and higher. This lets you specify the properties `keyStoreType` and `trustStoreType` in `oozie-site.xml`. OOZIE-3674 adds the parameter `--insecure` to the Oozie client so it can ignore certificate errors.

Oozie enforces TLS hostname verification, which means that any certificate you use for in-transit encryption must meet hostname verification requirements. If the certificate doesn't meet the criteria, the cluster might get stuck at the `oozie share lib update` stage when Amazon EMR provisions the cluster. We recommend that you update your certificates to make sure they're compliant with hostname verification. However, if you can't update the certificates, you can disable SSL for Oozie by setting the `oozie.https.enabled` property to `false` in cluster configuration.

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release
EmbeddedOozieServer	oozie.https.port	11443	TLS	emr-7.3.0+
EmbeddedOozieServer	oozie.email.smtp.port	25	TLS	emr-7.3.0+

Hue

By default, Hue supports TLS when in-transit encryption is enabled in Amazon EMR clusters. For more information about Hue configurations, see Configure Hue with HTTPS / SSL.

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release
Hue	http_port	8888	TLS	emr-7.4.0+

Livy

By default, Livy supports TLS when in-transit encryption is enabled in Amazon EMR clusters. For more information about Livy configurations, see Enabling HTTPS with Apache Livy.

Starting with Amazon EMR 7.3.0, if you use in-transit encryption and Kerberos authentication, you can't use the Livy server for Spark applications that depend on the Hive metastore. This

issue is fixed in HIVE-16340 and is fully resolved in SPARK-44114 when the open-source Spark application can upgrade to Hive 3. In the meantime, you can work around this issue if you set `hive.metastore.use.SSL` to `false`. For more information, see Configure applications.

For more information, see enabling HTTPS with Apache Livy.

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release
livy-server	livy.server.port	8998	TLS	emr-7.4.0+

JupyterEnterpriseGateway

By default, Jupyter Enterprise Gateway supports TLS when in-transit encryption is enabled in Amazon EMR clusters. For more information about the Jupyter Enterprise Gateway configurations, see Securing Enterprise Gateway Server.

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release
jupyter_enterprise_gateway	c.EnterpriseGatewayApp.port	9547	TLS	emr-7.4.0+

JupyterHub

By default, JupyterHub supports TLS when in-transit encryption is enabled in Amazon EMR clusters. For more information, see Enabling SSL encryption in the JupyterHub documentation. It isn't recommended to disable encryption.

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release
jupyter_hub	c.JupyterHub.port	9443	TLS	emr-5.14.0+, emr-6.0.0+, emr-7.0.0+

Zeppelin

By default, Zeppelin supports TLS when you enable in-transit encryption in your EMR cluster. For more information about the Zeppelin configurations, see SSL Configuration in the Zeppelin documentation.

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release
zeppelin	zeppelin.server.ssl.port	8890	TLS	7.3.0+

Zookeeper

Amazon EMR sets `serverCnxnFactory` to `org.apache.zookeeper.server.NettyServerCnxnFactory` to enable TLS for the Zookeeper quorum and client communication.

`secureClientPort` specifies the port that listens to TLS connections. If the client doesn't support TLS connections to Zookeeper, clients can connect to the insecure port of 2181 specified in `clientPort`. You can override or disable these two ports.

Amazon EMR sets both `sslQuorum` and `admin.forceHttps` to `true` to enable TLS communication for the quorum and admin server. If you don't want in-transit encryption for the quorum and the admin server, you can disable those configurations. We recommend that you use the default configurations for maximum security.

For more information, see Encryption, Authentication, Authorization Options in the Zookeeper documentation.

Component	Endpoint	Port	In-Transit Encryption Mechanism	Supported from Release
Zookeeper Server	secureClientPort	2281	TLS	emr-7.4.0+
Zookeeper Server	Quorum Ports	There are 2: Followers use 2888 to connect to the leader. Leader election uses 3888	TLS	emr-7.4.0+
Zookeeper Server	admin.serverPort	8341	TLS	emr-7.4.0+

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
- Authenticating with identities
- Managing access using policies
- How Amazon EMR works with IAM
- Runtime roles for Amazon EMR steps

- Configure IAM service roles for Amazon EMR permissions to AWS services and resources
- Amazon EMR identity-based policy examples

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that 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. If you cannot access a feature in Amazon EMR, see Troubleshooting Amazon EMR identity and access.

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 service users 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.

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.

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

Regardless of the authentication method that you use, you might 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 Multi-factor authentication in the *AWS IAM Identity Center User Guide* and AWS Multi-factor authentication in IAM 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 don't 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 *IAM User Guide*.

Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For

information about IAM Identity Center, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

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 Use cases for IAM users 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. To temporarily assume an IAM role in the AWS Management Console, you can switch from a user to an IAM role (console). 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 Methods to assume a role in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **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 Create a role for a third-party identity provider (federation) 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 User Guide*.

- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **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 Cross account resource access in IAM in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.
  - **Service role** – 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 Create a role to delegate permissions to an AWS service in the *IAM User Guide*.
  - **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **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 Use an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) 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*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

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, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed 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 Choose between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific

resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer 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 an 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 Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – RCPs are JSON policies that you can use to set the maximum available permissions for resources in your accounts without updating the IAM policies attached

to each resource that you own. The RCP limits permissions for resources in member accounts and can impact the effective permissions for identities, including the AWS account root user, regardless of whether they belong to your organization. For more information about Organizations and RCPs, including a list of AWS services that support RCPs, see Resource control policies (RCPs) 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

Before you use IAM to manage access to Amazon EMR, learn what IAM features are available to use with Amazon EMR.

IAM features you can use with Amazon EMR

IAM feature	Amazon EMR support
Identity-based policies	Yes
Resource-based policies	Yes
Policy actions	Yes
Policy resources	Yes
Policy condition keys	Yes
ACLs	No
ABAC (tags in policies)	Yes

IAM feature	Amazon EMR support
Temporary credentials	Yes
Principal permissions	Yes
Service roles	No
Service-linked roles	Yes

To get a high-level view of how Amazon EMR and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for Amazon EMR

Supports identity-based policies: Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

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. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for Amazon EMR

To view examples of Amazon EMR identity-based policies, see Amazon EMR identity-based policy examples.

Resource-based policies within Amazon EMR

Supports resource-based policies: Yes

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific

resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Policy actions for Amazon EMR

Supports policy actions: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of Amazon EMR actions, see Actions, resources, and condition keys for Amazon EMR in the *Service Authorization Reference*.

Policy actions in Amazon EMR use the following prefix before the action:

```
EMR
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
      "EMR:action1",
```

```
        "EMR:action2"
           ]
```

To view examples of Amazon EMR identity-based policies, see Amazon EMR identity-based policy examples.

Policy resources for Amazon EMR

Supports policy resources: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To see a list of Amazon EMR resource types and their ARNs, see Resources Defined by Amazon EMR in the *Service Authorization Reference*. To learn which actions you can specify the ARN of each resource, see Actions, resources, and condition keys for Amazon EMR.

To view examples of Amazon EMR identity-based policies, see Amazon EMR identity-based policy examples.

Policy condition keys for Amazon EMR

Supports service-specific policy condition keys: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create 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*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of Amazon EMR condition keys and to learn which actions and resources you can use a condition key, see Actions, resources, and condition keys for Amazon EMR in the *Service Authorization Reference*.

To view examples of Amazon EMR identity-based policies, see Amazon EMR identity-based policy examples.

Access control lists (ACLs) in Amazon EMR

Supports ACLs: No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Attribute-based access control (ABAC) with Amazon EMR

Supports ABAC (tags in policies)	Yes

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see Define permissions with ABAC authorization in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Using Temporary credentials with Amazon EMR

Supports temporary credentials: Yes

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see AWS services that work with IAM in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switch from a user to an IAM role (console) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

Cross-service principal permissions for Amazon EMR

Supports forward access sessions (FAS): Yes

When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a

different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

Service roles for Amazon EMR

Supports service roles	No

Service-linked roles for Amazon EMR

Supports service-linked roles	Yes

For details about creating or managing service-linked roles, see AWS services that work with IAM. Find a service in the table that includes a `Yes` in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

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.
- 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 see a list of Amazon EMR actions, see Actions Defined by Amazon EMR in the *IAM User Guide*.

Runtime roles for Amazon EMR steps

A *runtime role* is an AWS Identity and Access Management (IAM) role that you can specify when you submit a job or query to an Amazon EMR cluster. The job or query that you submit to your Amazon EMR cluster uses the runtime role to access AWS resources, such as objects in Amazon S3. You can specify runtime roles with Amazon EMR for Spark and Hive jobs.

You can also specify runtime roles when you connect to Amazon EMR clusters in Amazon SageMaker AI and when you attach an Amazon EMR Studio Workspace to an EMR cluster. For more information, see Connect to an Amazon EMR cluster from SageMaker AI Studio and Run an EMR Studio Workspace with a runtime role.

Previously, Amazon EMR clusters ran Amazon EMR jobs or queries with permissions based on the IAM policy attached to the instance profile that you used to launch the cluster. This meant that the policies had to contain the union of all the permissions for all jobs and queries that ran on an Amazon EMR cluster. With runtime roles, you can now manage access control for each job or query individually, instead of sharing the Amazon EMR instance profile of the cluster.

On Amazon EMR clusters with runtime roles, you can also apply AWS Lake Formation based access control to Spark, Hive, and Presto jobs and queries against your data lakes. To learn more on how to integrate with AWS Lake Formation, see Integrate Amazon EMR with AWS Lake Formation.

> **Note**
>
> When you specify a runtime role for an Amazon EMR step, the jobs or queries that you submit can only access AWS resources that the policies attached to the runtime role allow. These jobs and queries can't access the Instance Metadata Service on the EC2 instances of the cluster or use the EC2 instance profile of the cluster to access any AWS resources.

Prerequisites for launching an Amazon EMR cluster with a runtime role

Topics

- Step 1: Set up security configurations in Amazon EMR
- Step 2: Set up an EC2 instance profile for the Amazon EMR cluster
- Step 3: Set up a trust policy

Step 1: Set up security configurations in Amazon EMR

Use the following JSON structure to create a security configuration on the AWS Command Line Interface (AWS CLI), and set `EnableApplicationScopedIAMRole` to `true`. For more information about security configurations, see Use security configurations to set up Amazon EMR cluster security.

```
{
    "AuthorizationConfiguration":{
        "IAMConfiguration":{
            "EnableApplicationScopedIAMRole":true
        }
    }
}
```

We recommend that you always enable the in-transit encryption options in the security configuration, so that data transferred over the internet is encrypted, rather than in plain text. You can skip these options if you don’t want to connect to Amazon EMR clusters with runtime roles from SageMaker Runtime Studio or EMR Studio. To configure data encryption, see Configure data encryption.

Alternatively, you can create a security configuration with custom settings with the AWS Management Console.

Step 2: Set up an EC2 instance profile for the Amazon EMR cluster

Amazon EMR clusters use the Amazon EC2 instance profile role to assume the runtime roles. 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. To add policies to an IAM role or edit an existing inline or managed policy, see Adding and removing IAM identity permissions.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowRuntimeRoleUsage",
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ],
      "Resource": [
        "arn:aws:iam::123456789012:role/EMRRuntimeRole"
      ]
    }
  ]
}
```

Step 3: Set up a trust policy

For each IAM role that you plan to use as a runtime role, set the following trust policy, replacing `EMR_EC2_DefaultRole` with your instance profile role. To modify the trust policy of an IAM role, 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",
             "sts:TagSession"
```

```
            ]
}
```

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. To use runtime roles with Amazon EMR steps, use release label `emr-6.7.0` or later, and select Hive, Spark, or both as your cluster application. CloudWatchAgent is supported on Runtime Role Clusters for EMR 7.6 and above. To connect from SageMaker AI Studio, use release `emr-6.9.0` or later, and select Livy, Spark, Hive, or Presto as your cluster application. For instructions on how to launch your cluster, see Specify a security configuration for an Amazon EMR cluster.

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 $CLUSTER_ID \
--steps '[{ "Name": "Spark Example", "ActionOnFailure": "CONTINUE", "Jar":"command-
runner.jar","Args" : ["spark-example","HdfsTest", "$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 Amazon 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 Amazon 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", "Jar": "command-runner.jar","Args" :["hive -
f","s3://DOC_EXAMPLE_BUCKET/QUERY_FILE.hql"] }]' \
--execution-role-arn $RUNTIME_ROLE_ARN \
--region $REGION
```

Connect to Amazon EMR clusters with runtime roles from a SageMaker AI Studio notebook

You can apply Amazon EMR runtime roles to queries that you run in Amazon EMR clusters from SageMaker AI Studio. To do so, go through the following steps.

1. Follow the instructions in Launch Amazon SageMaker AI Studio to create an SageMaker AI Studio.
2. In the SageMaker AI Studio UI, start a notebook with supported kernels. For example, start a SparkMagic image with a PySpark kernel.
3. Choose an Amazon EMR cluster in SageMaker AI Studio, and then choose **Connect**.
4. Choose a runtime role, and then choose **Connect**.

This will create an SageMaker AI notebook cell with magic commands to connect to your Amazon EMR cluster with the chosen Amazon EMR runtime role. In the notebook cell, you can enter and run queries with runtime role and Lake Formation based access control. For a more detailed example, see Apply fine-grained data access controls with AWS Lake Formation and Amazon EMR from Amazon SageMaker AI Studio.

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.

> **Important**
>
> You must create a condition based on the `elasticmapreduce:ExecutionRoleArn` context key when you grant a caller access to call the `AddJobFlowSteps` or `GetClusterSessionCredentials` APIs, 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*"
            ]
        }
    }
}
```

Establish trust between runtime roles and Amazon EMR clusters

Amazon EMR generates a unique identifier `ExternalId` for each security configuration with activated runtime role authorization. This authorization allows every user to own a set of runtime roles to use on clusters that belong to them. For example, in an enterprise, every department can use their external ID to update the trust policy on their own set of runtime roles.

You can find the external ID with the Amazon EMR `DescribeSecurityConfiguration` API, as shown in the following example.

```
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"
}
```

For information about how to use an external ID, see How to use an external ID when granting access to your AWS resources to a third party.

Audit

To monitor and control actions that end users take with IAM roles, you can turn on the source identity feature. To learn more about source identity, see Monitor and control actions taken with assumed roles.

To track source identity, set `ApplicationScopedIAMRoleConfiguration/PropagateSourceIdentity` to `true` in your security configuration, as follows.

```
{
    "AuthorizationConfiguration":{
        "IAMConfiguration":{
            "EnableApplicationScopedIAMRole":true,
            "ApplicationScopedIAMRoleConfiguration":{
                "PropagateSourceIdentity":true
            }
        }
    }
}
```

When you set `PropagateSourceIdentity` to `true`, Amazon EMR applies the source identity from the calling credentials to a job or query session that you create with the runtime role. If no source identity is present in the calling credentials, Amazon EMR doesn't set the source identity.

To use this property, provide `sts:SetSourceIdentity` permissions to your instance profile, as follows.

```
{ // PropagateSourceIdentity statement
```

```
    "Sid":"PropagateSourceIdentity",
    "Effect":"Allow",
    "Action":"sts:SetSourceIdentity",
    "Resource":[
        <runtime-role-ARN>
    ],
    "Condition":{
        "StringEquals":{
            "sts:SourceIdentity":<source-identity>
        }
    }
}
```

You must also add the `AllowSetSourceIdentity` statement to the trust policy of your runtime roles.

```
{ // AllowSetSourceIdentity statement
    "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>
        }
    }
}
```

Additional considerations

> **Note**
>
> With Amazon EMR release `emr-6.9.0`, you might experience intermittent failures when you connect to Amazon EMR clusters from SageMaker AI Studio. To address this issue, you can install the patch with a bootstrap action when you launch the cluster. For patch details, see Amazon EMR release 6.9.0 known issues.

Additionally, consider the following when you configure runtime roles for Amazon EMR.

- Amazon EMR supports runtime roles in all commercial AWS Regions.
- Amazon EMR steps support Apache Spark and Apache Hive jobs with runtime roles when you use release `emr-6.7.0` or later.
- SageMaker AI Studio supports Spark, Hive, and Presto queries with runtime roles when you use release `emr-6.9.0` or later.
- The following notebook kernels in SageMaker AI support runtime roles:
  - DataScience – Python 3 kernel
  - DataScience 2.0 – Python 3 kernel
  - DataScience 3.0 – Python 3 kernel
  - SparkAnalytics 1.0 – SparkMagic and PySpark kernels
  - SparkAnalytics 2.0 – SparkMagic and PySpark kernels
  - SparkMagic – PySpark kernel
- Amazon EMR supports steps that use `RunJobFlow` only at the time of cluster creation. This API doesn't support runtime roles.
- Amazon EMR doesn't support runtime roles on clusters that you configure to be highly-available.
- Starting with Amazon EMR release 7.5.0 and higher, runtime roles support viewing Spark and YARN User Interfaces (UIs), such as the following: Spark Live UI, Spark History Server, YARN NodeManager, and YARN ResourceManager. When you navigate to these UIs, there is a username and password prompt. Usernames and passwords can be generated through use of the EMR GetClusterSessionCredentials API. For more information regarding usage details for the API, see GetClusterSessionCredentials.

  An example of how to use the EMR GetClusterSessionCredentials API is the following:

  ```
  aws emr  get-cluster-session-credentials --cluster-id <cluster_ID> --execution-role-
  arn <IAM_role_arn>
  ```

- You must escape your Bash command arguments when running commands with the `command-runner.jar` JAR file:

  ```
  aws emr add-steps --cluster-id <cluster-id> --steps '[{"Name":"sample-
  step","ActionOnFailure":"CONTINUE","Jar":"command-runner.jar","Properties":"","Args":
  ["bash","-c","\"aws s3 ls\""],"Type":"CUSTOM_JAR"}]' --execution-role-
  arn <IAM_ROLE_ARN>
  ```

Additionally, you must escape your Bash command arguments when running commands with the script runner. The following is a sample that shows setting Spark properties, with included escape characters:

```
"\"--conf spark.sql.autoBroadcastJoinThreshold=-1\n--conf
 spark.cradle.RSv2Mode.enabled=true\""
```

- Runtime roles don't provide support for controlling access to on-cluster resources, such as HDFS and HMS.
- Runtime roles don't provide support for docker/containers.

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 with Amazon EMR.

Modifying identity-based policies for permissions to pass service roles for Amazon EMR

The Amazon EMR full-permissions default managed policies incorporate `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 with the 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)	`EMR_DefaultRole_V2`	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.	`AmazonEMRServicePolicy_v2` **Important** A service-linked role is required to request Spot Instances. If this role doesn't exist, the Amazon EMR service role must have

Function	Default role	Description	Default managed policy
			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) and Service-linked role for Spot Instance requests in the *Amazon EC2 User Guide*.

Function	Default role	Description	Default managed policy
Service role for cluster EC2 instances (EC2 instance profile)	`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 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. This role is required for all clusters.	`AmazonElasticMapReduceforEC2Role` . For more information, see Service role for cluster EC2 instances (EC2 instance profile).

Function	Default role	Description	Default managed policy
Service role for automatic scaling in Amazon EMR (Auto Scaling role)	`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 in Amazon EMR.	`AmazonElasticMapReduceforAutoScalingRole` . For more information, see Service role for automatic scaling in Amazon EMR (Auto Scaling role).

Function	Default role	Description	Default managed policy
Service role for EMR Notebooks	`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.	`AmazonElasticMapReduceEditorsRole` . For more information, see Service role for EMR Notebooks. `S3FullAccessPolicy` is also attached by default. Following is the contents of this policy. JSON `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "s3:*" ], "Resource": [`

Function	Default role	Description	Default managed policy
			```"*"], "Sid": "AllowS3" }] }```
Service-Linked Role	`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) must allow the creation of a service-linked role. For more information, see Using service-linked roles for Amazon EMR.	`AmazonEMRCleanupPolicy`

Topics

- IAM service roles used by Amazon EMR
- Customize IAM roles with Amazon EMR
- Configure IAM roles for EMRFS requests to Amazon S3
- Use resource-based policies for Amazon EMR access to AWS Glue Data Catalog
- Use IAM roles with applications that call AWS services directly
- Allow users and groups to create and modify roles

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.

Topics

- Service role for Amazon EMR (EMR role)
- Service role for cluster EC2 instances (EC2 instance profile)
- Service role for automatic scaling in Amazon EMR (Auto Scaling role)
- Service role for EMR Notebooks
- Using service-linked roles for Amazon EMR

Service role for Amazon EMR (EMR role)

The Amazon EMR role defines the allowable actions for Amazon EMR when it provisions resources and performs service-level tasks that aren't performed in the context of an Amazon 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_V2`.

- The Amazon EMR scoped default managed policy attached to `EMR_DefaultRole_V2` is `AmazonEMRServicePolicy_v2`. This v2 policy replaces the deprecated default managed policy, `AmazonElasticMapReduceRole`.

`AmazonEMRServicePolicy_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 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 Amazon EMR. See Tagging resources to use managed policies.

> **Important**
>
> Amazon EMR uses this Amazon EMR service role and the `AWSServiceRoleForEMRCleanup` role to clean up cluster resources in your account that you no longer use, such as Amazon EC2 instances. You must include actions for the role policies to delete or terminate the resources. Otherwise, Amazon EMR can't perform these cleanup actions, and you might incur costs for unused resources that remain on the cluster.

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.

JSON

```
{
  "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:DescribeInstanceTypeOfferings",
```

```
        "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*"
        }
      }
    },
    {
      "Sid": "CreateAndModifyEmrServiceVPCEndpoint",
      "Effect": "Allow",
      "Action": [
        "ec2:ModifyVpcEndpoint",
        "ec2:CreateVpcEndpoint"
      ],
      "Resource": [
```

```
        "arn:aws:ec2:*:*:vpc-endpoint/*",
        "arn:aws:ec2:*:*:subnet/*",
        "arn:aws:ec2:*:*:security-group/*",
        "arn:aws:ec2:*:*:vpc/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/for-use-with-amazon-emr-managed-policies": "true"
        }
      }
    },
    {
      "Sid": "CreateEmrServiceVPCEndpoint",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateVpcEndpoint"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:vpc-endpoint/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/for-use-with-amazon-emr-managed-policies": "true",
          "aws:RequestTag/Name": "emr-service-vpce"
        }
      }
    },
    {
      "Sid": "TagEmrServiceVPCEndpoint",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateTags"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:vpc-endpoint/*"
      ],
      "Condition": {
        "StringEquals": {
          "ec2:CreateAction": "CreateVpcEndpoint",
          "aws:RequestTag/for-use-with-amazon-emr-managed-policies": "true",
          "aws:RequestTag/Name": "emr-service-vpce"
        }
      }
    }
```

```
  ]
}
```

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.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSTSAssumerole",
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": "arn:aws:iam::123456789012:role/EMRServiceRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:elasticmapreduce:*:123456789012:*"
        }
      }
    }
  ]
}
```

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 nearing end of support. 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.

The following shows the contents of version 3 of `AmazonElasticMapReduceforEC2Role`.

JSON

```
{
  "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"
      ],
      "Sid": "AllowCLOUDWATCH"
    }
  ]
}
```

Your service role should use the following trust policy.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSTSAssumerole",
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": "arn:aws:iam::123456789012:role/EMR_EC2_DefaultRole"
    }
  ]
}
```

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 with Amazon EMR.

> **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.

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.

JSON

```
{
  "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/*"
      ],
      "Sid": "AllowS3Abortmultipartupload"
    },
    {
      "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:*:123456789012:table/EmrFSMetadata"
      ],
      "Sid": "AllowDYNAMODBCreatetable"
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:PutMetricData",
        "dynamodb:ListTables",
        "s3:ListBucket"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowCLOUDWATCHPutmetricdata"
    },
    {
      "Effect": "Allow",
      "Action": [
        "sqs:GetQueueUrl",
        "sqs:ReceiveMessage",
        "sqs:DeleteQueue",
        "sqs:SendMessage",
        "sqs:CreateQueue"
      ],
      "Resource": [
        "arn:aws:sqs:*:123456789012:EMRFS-Inconsistency-*"
      ],
      "Sid": "AllowSQSGetqueueurl"
    }
  ]
}
```

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.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::MyLoggingBucket/MyEMRClusterLogs/*"
      ],
      "Sid": "AllowS3Putobject"
    }
  ]
}
```

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*.

JSON

```
{
  "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": [
        "*"
      ],
      "Sid": "AllowGLUECreatedatabase"
    }
  ]
}
```

Service role for automatic scaling in Amazon EMR (Auto Scaling role)

The Auto Scaling role for Amazon 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.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "cloudwatch:DescribeAlarms",
        "elasticmapreduce:ListInstanceGroups",
        "elasticmapreduce:ModifyInstanceGroups"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ],
      "Sid": "AllowCLOUDWATCHDescribealarms"
    }
  ]
}
```

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.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
```

```
      ],
      "Resource": "arn:aws:iam::123456789012:role/ApplicationAutoScalingEMRRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:application-
autoscaling:*:123456789012:scalable-target/*"
        }
      },
      "Sid": "AllowSTSAssumerole"
    }
  ]
}
```

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.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": "arn:aws:iam::123456789012:role/EMRServiceRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:elasticmapreduce:*:123456789012:*"
        }
      },
      "Sid": "AllowSTSAssumerole"
    }
  ]
}
```

The contents of version 1 of `AmazonElasticMapReduceEditorsRole` are as follows.

JSON

```
{
  "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": [
        "*"
      ],
      "Sid": "AllowEC2Authorizesecuritygroupegress"
    },
    {
      "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"
          ]
        }
      },
      "Sid": "AllowEC2Createtags"
    }
  ]
}
```

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.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:*"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowS3"
    }
  ]
}
```

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 Amazon EMR notebooks. An example policy is demonstrated below:

JSON

```
{
  "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 that are 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",`

Action	Permissions

	`"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"`

EMR Notebooks updates to AWS managed policies

View details about updates to AWS managed policies for EMR Notebooks since March 1, 2021.

Change	Description	Date
`AmazonElasticMapReduceEditorsRole - Added permissions`	EMR Notebooks added `ec2:describeVPCs` and `elastmicmapreduce:ListSteps` permissions to `AmazonElasticMapReduceEditorsRole` .	Feb 8, 2023
EMR Notebooks started tracking changes	EMR Notebooks started tracking changes for its AWS managed policies.	Feb 8, 2023

Using service-linked roles 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. Service-linked roles are predefined by Amazon EMR and include all the permissions that the service requires to call other AWS services on your behalf.

Topics

- Using service-linked roles for Amazon EMR for cleanup
- Using service-linked roles with Amazon EMR for write-ahead logging

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 roles** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Using service-linked roles for Amazon EMR for cleanup

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. Service-linked roles are predefined by Amazon EMR and include all the permissions that the service requires to call other AWS services on your behalf.

Service-linked roles work 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.

A service-linked role makes setting up Amazon EMR easier because you don't have to manually add the necessary permissions. Amazon EMR defines the permissions of its service-linked roles, and unless defined otherwise, only Amazon EMR can assume its roles. 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 this service-linked role for Amazon EMR only after you delete any related resources and terminate all EMR clusters in the account. This protects your Amazon EMR resources so that you can't inadvertently remove permission to access the resources.

Using service-linked roles for cleanup

Amazon EMR uses the service-based **AWSServiceRoleForEMRCleanup** role to grant Amazon EMR permission to terminate and delete Amazon EC2 resources on your behalf if the Amazon EMR service-linked role loses that capability. Amazon EMR creates the service-linked role automatically during cluster creation if it doesn't 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: `DescribeLaunchTemplates` on `ec2`
- Action: `DeleteLaunchTemplate` 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`
- Action: `DescribePlacementGroups` on `ec2`
- Action: `DeletePlacementGroup` 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.

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 the AWSServiceRoleForEMRCleanup service-linked role is not present, Amazon EMR creates the AWSServiceRoleForEMRCleanup service-linked role for you. You must have permissions to create a 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):

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"
   ],
   "Resource": "arn:aws:iam::*:role/aws-service-role/elasticmapreduce.amazonaws.com*/
AWSServiceRoleForEMRCleanup*",
   "Condition": {"StringLike": {"iam:AWSServiceName": "ops.emr-
serverless.amazonaws.com"}}
 }
```

> **Important**
>
> If you used Amazon EMR before October 24, 2017, when service-linked roles weren't supported, then Amazon EMR created the AWSServiceRoleForEMRCleanup service-linked 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 doesn't allow you to edit the AWSServiceRoleForEMRCleanup service-linked role. After you create a service-linked role, you can't change the name of the service-linked role because various entities might reference the service-linked role. However, you can edit the description of the service-linked 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 service-linked 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 service-linked role has no active sessions and remove any resources used by the service-linked 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 service-linked role.
3. On the **Summary** page for the selected service-linked 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 service-linked role, you can try to delete the service-linked role. If the service is using the service-linked role, then the deletion fails and you can view the Regions where the service-linked role is being used. If the service-linked role is being used, then you must wait for the session to end before you can delete the service-linked 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 an Amazon EMR cluster in the starting, running, or waiting states.

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 service-linked 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 AWSServiceRoleForEMRCleanup

Amazon EMR supports using the AWSServiceRoleForEMRCleanup service-linked role 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
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

Using service-linked roles with Amazon EMR for write-ahead logging

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. Service-linked roles are predefined by Amazon EMR and include all the permissions that the service requires to call other AWS services on your behalf.

Service-linked roles work 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.

A service-linked role makes setting up Amazon EMR easier because you don't have to manually add the necessary permissions. Amazon EMR defines the permissions of its service-linked roles, and unless defined otherwise, only Amazon EMR can assume its roles. 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 this service-linked role for Amazon EMR only after you delete their related resources and terminate all EMR clusters in the account. This protects your Amazon EMR resources so that you can't inadvertently remove permission to access the resources.

Service-linked role permissions for write-ahead logging (WAL)

Amazon EMR uses the service-linked role **AWSServiceRoleForEMRWAL** to retrieve a cluster status.

The AWSServiceRoleForEMRWAL service-linked role trusts the following services to assume the role:

- `emrwal.amazonaws.com`

The `EMRDescribeClusterPolicyForEMRWAL` permissions policy for the service-linked role allows Amazon EMR to complete the following actions on the specified resources:

- Action: `DescribeCluster` on `*`

You must configure permissions to allow an IAM entity (in this case, Amazon EMR WAL) to create, edit, or delete a service-linked role. Add the following statements as needed to the permissions policy for your instance profile:

CreateServiceLinkedRole

To allow an IAM entity to create the AWSServiceRoleForEMRWAL 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/emrwal.amazonaws.com*/
AWSServiceRoleForEMRWAL*",
    "Condition": {
        "StringLike": {
            "iam:AWSServiceName": [
                "emrwal.amazonaws.com",
                "elasticmapreduce.amazonaws.com.cn"
            ]
        }
    }
}
```

UpdateRoleDescription

To allow an IAM entity to edit the description of the AWSServiceRoleForEMRWAL 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/emrwal.amazonaws.com*/
AWSServiceRoleForEMRWAL*",
    "Condition": {
        "StringLike": {
```

```
            "iam:AWSServiceName": [
                "emrwal.amazonaws.com",
                "elasticmapreduce.amazonaws.com.cn"
            ]
        }
    }
}
```

DeleteServiceLinkedRole

To allow an IAM entity to delete the AWSServiceRoleForEMRWAL 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": [
                "emrwal.amazonaws.com",
                "elasticmapreduce.amazonaws.com.cn"
            ]
        }
    }
}
```

Creating a service-linked role for Amazon EMR

You don't need to manually create the AWSServiceRoleForEMRWAL role. Amazon EMR creates this service-linked role automatically when you create a WAL workspace with the EMRWAL CLI or from AWS CloudFormation, or HBase will create the service-linked role when you configure a workspace for Amazon EMR WAL and the service-linked role doesn't yet exist. You must have permissions to create a service-linked role. For example statements that add this capability to the permissions policy of an IAM entity (such as a user, group, or role), see the prior section, Service-linked role permissions for write-ahead logging (WAL).

Editing a service-linked role for Amazon EMR

Amazon EMR doesn't allow you to edit the AWSServiceRoleForEMRWAL service-linked role. After you create a service-linked role, you can't change the name of the service-linked role because various entities might reference the service-linked role. However, you can edit the description of the service-linked 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 service-linked 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.

> **Note**
>
> The write-ahead logging operation isn't affected if you delete the AWSServiceRoleForEMRWAL role, but Amazon EMR won't auto-delete the logs that it created once your EMR cluster terminates. Therefore, you'll need to manually delete the Amazon EMR WAL logs if you delete the service-linked role.

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 AWSServiceRoleForEMRWAL 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 AWSServiceRoleForEMRWAL role, you can try to delete the service-linked role. If the service is using the role, then the deletion fails and you can view the Regions where the service-linked role is being used. If the service-linked role is being used, then you must wait for the session to end before you can delete the service-linked role. You cannot revoke the session for a service-linked role.

To remove Amazon EMR resources used by the AWSServiceRoleForEMRWAL

- Terminate all clusters in your account. For more information, see Terminate an Amazon EMR cluster in the starting, running, or waiting states.

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 AWSServiceRoleForEMRWAL, 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 AWSServiceRoleForEMRWAL
   ```

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 AWSServiceRoleForEMRWAL 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 AWSServiceRoleForEMRWAL

Amazon EMR supports using the AWSServiceRoleForEMRWAL service-linked role 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 (Singapore)	ap-southeast-1	Yes
Asia Pacific (Sydney)	ap-southeast-2	Yes
Asia Pacific (Tokyo)	ap-northeast-1	Yes
Europe (Frankfurt)	eu-central-1	Yes
Europe (Ireland)	eu-west-1	Yes

Customize IAM roles with Amazon EMR

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.

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.

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 **account is not authorized to call EC2** error occurs. For more information, see Allow users and groups to create and modify roles.

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.

Console

To specify custom IAM roles with the console

When you create a cluster with the console, you must specify a custom service role for Amazon EMR and a custom role for the EC2 instance profile. For more information, see IAM service roles used by Amazon EMR.

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose **Create cluster**.
3. Under **Security configuration and permissions**, find the **IAM role for instance profile** and **Service role for Amazon EMR** fields. 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.
4. Choose any other options that apply to your cluster.
5. To launch your cluster, choose **Create cluster**.

Use the AWS CLI to specify custom roles

You can specify a service role for Amazon 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 in Amazon EMR.

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-7.10.0 \
--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

> **Note**
>
> The EMRFS role mapping capability described on this page has been improved upon with the introduction of Amazon S3 Access Grants in Amazon EMR 6.15.0. For a scalable access control solution for your data in Amazon S3, we recommend that you use S3 Access Grants with Amazon EMR.

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. 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).

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://amzn-s3-demo-bucket` or `s3://amzn-s3-demo-bucket/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).

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 Amazon 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).

Updating trust policies to assume role permissions

Each role that EMRFS uses must have a trust policy that allows the cluster's Amazon EMR role for EC2 to assume it. Similarly, the cluster's Amazon 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 Amazon 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.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": "arn:aws:iam::123456789012:role/EMR_EC2_DefaultRole",
      "Sid": "AllowSTSAssumerole"
    }
  ]
}
```

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.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": [
        "arn:aws:iam::123456789012:role/EMRFSRole_First",
        "arn:aws:iam::123456789012:role/EMRFSRole_Second"
      ],
      "Sid": "AllowSTSAssumerole"
    }
  ]
}
```

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 Amazon 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 with Amazon EMR and Specify custom IAM roles when you create a cluster.

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 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 with the Amazon EMR console or with the AWS CLI.
2. Specify the security configuration you created above when you create a cluster. For more information, see Specify a security configuration for an Amazon EMR cluster.

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 with the Amazon EMR console or with 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_demo_s3_buckets",
        "IdentifierType": "Prefix",
        "Identifiers": [ "s3://amzn-s3-demo-bucket1/","s3://amzn-s3-demo-
bucket2/" ]
      },{
        "Role": "arn:aws:iam::123456789101:role/allow_EMRFS_access_for_AdminGroup",
        "IdentifierType": "Group",
        "Identifiers": [ "AdminGroup" ]
      }]
    }
  }
}
```

Parameter	Description
`"AuthorizationConfiguration":`	Required.

Parameter	Description
`"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.

Parameter	Description
`"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://amzn-s3-demo-bucket/` matches `s3://amzn-s3-demo-bucket/mydir` and `s3://amzn-s3-demo-bucket/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.

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 Amazon 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-7.10.0 --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 a user, see Changing permissions for a user in the *IAM User Guide*.

Amazon EMR identity-based policy examples

By default, a 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 a 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
- Allow users to view their own permissions
- Amazon EMR managed policies
- IAM policies for tag-based access to clusters and EMR notebooks
- Denying the ModifyInstanceGroup action in Amazon EMR
- Troubleshooting Amazon EMR identity and access

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.
- **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 a 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 a 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.

JSON

```
{
  "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.

Amazon EMR managed policy changes

Policy type	Policy names	Policy purpose	Changes to v2 policy
Default EMR service role and attached managed policy	Role name: **EMR_DefaultRole**	Allows Amazon EMR to call other AWS services on your behalf	Policy adds the new permissio n `"ec2:Desc ribeInsta`

Policy type	Policy names	Policy purpose	Changes to v2 policy
	V1 policy (to be deprecated): **AmazonElasticMapReduceRole** (EMR Service Role) V2 (scoped-down) policy name: `AmazonEMRServicePolicy_v2`	when provisioning resources and performing service-level actions. This role is required for all clusters.	`nceTypeOfferings"` . This API operation returns a list of instance types that are supported by a list of given Availability Zones.
IAM managed policy for full Amazon EMR access by attached user, role, or group	V2 (scoped) policy name: `AmazonEMRServicePolicy_v2`	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. 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).

Policy type	Policy names	Policy purpose	Changes to v2 policy
IAM managed policy for read-only access by attached user, role, or group	V1 policy (to be deprecated): `AmazonElasticMapReduceReadOnlyAccess` V2 (scoped) policy name: `AmazonEMRReadOnlyAccessPolicy_v2`	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).
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`	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. See Service role for Amazon EMR (EMR role).

Policy type	Policy names	Policy purpose	Changes to v2 policy
Service role for cluster EC2 instances (EC2 instance profile)	Role name: **EMR_EC2_DefaultRole** Deprecated policy name: **AmazonElasticMapReduceforEC2Role**	Allows applications running on an EMR cluster to access other AWS resources , such as Amazon S3. For example, if you run Apache Spark jobs that process data from Amazon S3, the policy needs to allow access to such resources.	Both default role and default policy are on the path to deprecation. There is 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.
Other EC2 service role policies	Current policy names: **AmazonElasticMapReduceforAutoScalingRole, AmazonElasticMapReduceEditorsRole, AmazonEMRCleanupPolicy**	Provides permissions that Amazon EMR needs to access other 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 policies incorporate `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 with the 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 a 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 must pass the predefined user tag `for-use-with-amazon-emr-managed-policies = true` when you provision the cluster. Amazon EMR will then automatically propagate that tag. Additionally, you must add a user tag to the resources listed in the following section. If you use the Amazon EMR console to launch your cluster, see Considerations for using the Amazon EMR console to launch clusters with v2 managed policies.

To use managed policies, pass the user tag `for-use-with-amazon-emr-managed-policies = true` when you provision a cluster with 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 must add tags to those resources. For example, you must tag Amazon 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.

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 old 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 or if you want to use the console, ask your administrator to tag those resources.
- You cannot launch clusters with v2 managed policies unless all the prerequisites are met.
- The old Amazon EMR console shows you which resources (VPC/Subnets) need to be tagged.

IAM managed policy for full access (v2 managed default policy) for Amazon EMR

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` 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.

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.

JSON

```
{
  "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:CreatePersistentAppUI",
        "elasticmapreduce:CreateSecurityConfiguration",
        "elasticmapreduce:DeleteEditor",
        "elasticmapreduce:DeleteSecurityConfiguration",
        "elasticmapreduce:DescribeCluster",
        "elasticmapreduce:DescribeEditor",
        "elasticmapreduce:DescribeJobFlows",
        "elasticmapreduce:DescribePersistentAppUI",
        "elasticmapreduce:DescribeSecurityConfiguration",
        "elasticmapreduce:DescribeStep",
        "elasticmapreduce:DescribeReleaseLabel",
        "elasticmapreduce:GetBlockPublicAccessConfiguration",
        "elasticmapreduce:GetManagedScalingPolicy",
        "elasticmapreduce:GetPersistentAppUIPresignedURL",
        "elasticmapreduce:GetAutoTerminationPolicy",
        "elasticmapreduce:ListBootstrapActions",
        "elasticmapreduce:ListClusters",
        "elasticmapreduce:ListEditors",
        "elasticmapreduce:ListInstanceFleets",
        "elasticmapreduce:ListInstanceGroups",
        "elasticmapreduce:ListInstances",
        "elasticmapreduce:ListSecurityConfigurations",
        "elasticmapreduce:ListSteps",
        "elasticmapreduce:ListSupportedInstanceTypes",
        "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` and `AmazonEMRFullAccessPolicy_v2` AWS Identity and Access Management (IAM) managed policies grant all the required actions for Amazon EMR and other services.

> **Important**
>
> The `AmazonElasticMapReduceFullAccess` managed policy is on the path to deprecation, and no longer recommended for use with Amazon EMR. Instead, use `AmazonEMRFullAccessPolicy_v2`. When the IAM service eventually deprecates the v1 policy, you won't be able to attach it to a role. However, you can attach an existing role to a cluster even if that role uses the deprecated policy.

The Amazon EMR full-permissions default managed policies incorporate `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 with the v2 managed policies.

You can view the contents of the deprecated v1 policy in the AWS Management Console at `AmazonElasticMapReduceFullAccess`. The `ec2:TerminateInstances` action in the policy grants permission to the a user or role to terminate any of the Amazon EC2 instances associated with the IAM account. This includes instances that are not part of an EMR cluster.

IAM managed policy for read-only access (v2 managed default policy) for Amazon EMR

To grant read-only privileges to Amazon EMR, attach the **AmazonEMRReadOnlyAccessPolicy_v2** managed policy. This default managed policy replaces the `AmazonElasticMapReduceReadOnlyAccess` 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 allowable `elasticmapreduce` actions.

> **Note**
>
> You can also use the AWS Management Console link `AmazonEMRReadOnlyAccessPolicy_v2` to view the policy.

JSON

```
{
  "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:ListSupportedInstanceTypes",
        "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` 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.

JSON

```
{
  "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": [
        "*"
      ],
      "Sid": "AllowELASTICMAPREDUCEDescribe"
    }
  ]
}
```

AWS managed policy: EMRDescribeClusterPolicyForEMRWAL

You can't attach `EMRDescribeClusterPolicyForEMRWAL` to your IAM entities. This policy is attached to a service-linked role that allows Amazon EMR to perform actions on your behalf. For

more information on this service-linked role, see Using service-linked roles with Amazon EMR for write-ahead logging.

This policy grants read-only permissions that allow the WAL service for Amazon EMR to find and return the status of a cluster. For more information about Amazon EMR WAL, see Write-ahead logs (WAL) for Amazon EMR.

Permissions details

This policy includes the following permissions:

- `emr` – Allows principals to describe cluster status from Amazon EMR. This is required so that Amazon EMR can confirm when a cluster has terminated and then, after thirty days, clean up any WAL logs left behind by the cluster.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:DescribeCluster"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowELASTICMAPREDUCEDescribecluster"
    }
  ]
}
```

AWS managed policies for Amazon EMR

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies 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.

Change	Description	Date
`AmazonEMRServicePolicy_v2` – Update to an existing policy	Added `ec2:CreateVpcEndpoint`, `ec2:ModifyVpcEndpoint`, and `ec2:CreateTags` required for optimal experience, starting with Amazon EMR release 7.5.0.	March 4, 2025
`AmazonEMRServicePolicy_v2` – Update to an existing policy	Added `elasticmapreduce:CreatePersistentAppUI`, `elasticmapreduce:DescribePersistentAppUI`, and `elasticmapreduce:GetPersistentAppUIPresignedURL`.	February 28, 2025

Change	Description	Date
EMRDescribeClusterPolicyForEMRWAL – New policy	Added a new policy so that Amazon EMR can determine cluster status for WAL cleanup thirty days after cluster termination.	August 10, 2023
AmazonEMRFullAccessPolicy_v2 and **AmazonEMRReadOnlyAccessPolicy_v2** – Update to an existing policy	Added `elasticmapreduce:DescribeReleaseLabel` and `elasticmapreduce:GetAutoTerminationPolicy`.	April 21, 2022
AmazonEMRFullAccessPolicy_v2 – Update to an existing policy	Added `ec2:DescribeImages` for Using a custom AMI to provide more flexibility for Amazon EMR cluster configuration.	February 15, 2022
Amazon EMR managed policies	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

Change	Description	Date
AmazonEMRFullAccessPolicy_v2 – 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 `"iam:PassedToService":"ec2.amazonaws.com*"`, respectively.	May 20, 2021
AmazonEMRFullAccessPolicy_v2 – 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

Change	Description	Date
`AmazonEMRFullAccessPolicy_v2` – 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` – New policy	Adds a prerequisite that users must add user tags to resources before they can use this policy.	March 11, 2021
`AmazonEMRReadOnlyAccessPolicy_v2` – 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.

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 you don't deny tagging actions for a resource, a user can modify tags and circumvent the intention of the tag-based policies.

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 in Amazon EMR.

Topics

- Allow actions only on clusters with specific tag values
- Require cluster tagging when a cluster is created
- Allow actions on clusters with a specific tag, regardless of tag value

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.

JSON

```
{
  "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 prior example, the following policy example looks for the same matching tag: the value *dev* for the *department* tag. But in this example, the `RequestTag` condition key specifies that the policy applies during tag creation. So you must create a cluster with a tag that matches the specified value.

To create a cluster with a tag, you must also have permission for the `elasticmapredue:AddTags` action. For this statement, the `elasticmapreduce:ResourceTag` condition key ensures that IAM only grants access to tag resources with the value *dev* on the *department* tag. The `Resource` element is used to limit this permission to cluster resources.

For the `PassRole` resources, you must provide the AWS account ID or alias, the service role name in the `PassRoleForEMR` statement, and the instance profile name in the `PassRoleForEC2` statement. For more information about the IAM ARN format, see IAM ARNs in the *IAM User Guide*.

For more information about matching tag-key values, see `aws:RequestTag/tag-key` in the *IAM User Guide*.

JSON

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

```
        "Effect": "Allow",
        "Action": [
          "elasticmapreduce:RunJobFlow"
        ],
        "Resource": [
          "*"
        ],
        "Condition": {
          "StringEquals": {
            "aws:RequestTag/department": "dev"
          }
        }
      },
      {
        "Sid": "AddTagsForDevClusters",
        "Effect": "Allow",
        "Action": [
          "elasticmapreduce:AddTags"
        ],
        "Resource": [
          "arn:aws:elasticmapreduce:*:*:cluster/*"
        ],
        "Condition": {
          "StringEquals": {
            "elasticmapreduce:ResourceTag/department": "dev"
          }
        }
      },
      {
        "Sid": "PassRoleForEMR",
        "Effect": "Allow",
        "Action": [
          "iam:PassRole"
        ],
        "Resource": [
          "arn:aws:iam::123456789012:role/Role-Name-With-Path"
        ],
        "Condition": {
          "StringLike": {
            "iam:PassedToService": "elasticmapreduce.amazonaws.com*"
          }
        }
      },
      {
```

```
      "Sid": "PassRoleForEC2",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::123456789012:role/Role-Name-With-Path"
      ],
      "Condition": {
        "StringLike": {
          "iam:PassedToService": "ec2.amazonaws.com*"
        }
      }
    }
  ]
}
```

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*.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateClusterTagNullCondition",
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:RunJobFlow"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "Null": {
          "aws:RequestTag/department": "false"
        }
      }
    },
    {
      "Sid": "AddTagsNullCondition",
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:AddTags"
      ],
      "Resource": [
        "arn:aws:elasticmapreduce:*:*:cluster/*"
      ],
      "Condition": {
        "Null": {
          "elasticmapreduce:ResourceTag/department": "false"
        }
      }
    },
    {
      "Sid": "PassRoleForElasticMapReduce",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
```

```
      ],
      "Resource": [
        "arn:aws:iam::123456789012:role/Role-Name-With-Path"
      ],
      "Condition": {
        "StringLike": {
          "iam:PassedToService": "elasticmapreduce.amazonaws.com*"
        }
      }
    },
    {
      "Sid": "PassRoleForEC2",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::123456789012:role/Role-Name-With-Path"
      ],
      "Condition": {
        "StringLike": {
          "iam:PassedToService": "ec2.amazonaws.com*"
        }
      }
    }
  ]
}
```

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 a 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 a 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.

JSON

```
{
  "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}"
        }
      },
      "Sid": "AllowELASTICMAPREDUCEDescribeeditor"
    }
  ]
}
```

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.

JSON

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

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`.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "elasticmapreduce:CreateEditor"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ],
      "Condition": {
```

```
        "StringEquals": {
          "elasticmapreduce:RequestTag/dept": [
            "datascience",
            "analytics",
            "operations"
          ]
        }
      },
      "Sid": "AllowELASTICMAPREDUCECreateeditor"
    }
  ]
}
```

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.

JSON

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

```
            "dep3"
          ]
        }
      },
      "Sid": "AllowELASTICMAPREDUCECreateeditor"
    }
  ]
}
```

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.

JSON

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

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

JSON

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

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

JSON

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

```
}
```

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.

JSON

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

```
      ],
      "Condition": {
        "StringEquals": {
          "elasticmapreduce:ResourceTag/department": [
            "datascience",
            "analytics"
          ]
        }
      },
      "Sid": "AllowELASTICMAPREDUCEOpeneditorconsoleByDepartment"
    }
  ]
}
```

Denying the ModifyInstanceGroup action in Amazon EMR

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.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "elasticmapreduce:ModifyInstanceGroups"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ],
      "Sid": "AllowELASTICMAPREDUCEModifyinstancegroups"
    },
    {
      "Action": [
        "elasticmapreduce:ModifyInstanceGroups"
      ],
      "Effect": "Deny",
```

```
      "Resource": [
        "arn:aws:elasticmapreduce:us-east-1:123456789012:cluster/
j-12345ABCDEFG67"
      ],
      "Sid": "DenyELASTICMAPREDUCEModifyinstancegroups"
    }
  ]
}
```

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.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "elasticmapreduce:ModifyInstanceGroups"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ],
      "Sid": "AllowELASTICMAPREDUCEModifyinstancegroups"
    },
    {
      "Action": [
        "elasticmapreduce:ModifyInstanceGroups"
      ],
      "Effect": "Deny",
      "Resource": [
```

```
        "arn:aws:elasticmapreduce:us-east-1:123456789012:cluster/
j-12345ABCDEFG67"
      ],
      "Sid": "DenyELASTICMAPREDUCEModifyinstancegroupsSpecificCluster"
    },
    {
      "Action": [
        "elasticmapreduce:ModifyInstanceGroups"
      ],
      "Effect": "Deny",
      "NotResource": "arn:*:elasticmapreduce:*:*:cluster/*",
      "Sid": "DenyELASTICMAPREDUCEModifyinstancegroupsNonCluster"
    }
  ]
}
```

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.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "elasticmapreduce:ModifyInstanceGroups"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ],
      "Sid": "AllowELASTICMAPREDUCEModifyinstancegroups"
    },
```

```
    {
      "Action": [
        "elasticmapreduce:ModifyInstanceGroups"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/department": "dev"
        }
      },
      "Effect": "Deny",
      "Resource": [
        "*"
      ],
      "Sid": "DenyELASTICMAPREDUCEModifyinstancegroupsDevDepartment"
    },
    {
      "Action": [
        "elasticmapreduce:ModifyInstanceGroups"
      ],
      "Condition": {
        "StringNotLike": {
          "aws:ResourceTag/department": "?*"
        }
      },
      "Effect": "Deny",
      "Resource": [
        "*"
      ],
      "Sid": "DenyELASTICMAPREDUCEModifyinstancegroupsNoDepartmentTag"
    }
  ]
}
```

Troubleshooting Amazon EMR identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Amazon EMR and IAM.

Topics

- I am not authorized to perform an action in Amazon EMR
- I am not authorized to perform iam:PassRole

- I want to allow people outside of my AWS account to access my Amazon EMR resources

I am not authorized to perform an action in Amazon EMR

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

The following example error occurs when the `mateojackson` user tries to use the console to view details about a fictional *`my-example-widget`* resource but does not have the fictional `EMR:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 EMR:GetWidget on resource: my-example-widget
```

In this case, Mateo asks his administrator to update his policies to allow him to access the *`my-example-widget`* resource using the `EMR:`*`GetWidget`* action.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to Amazon EMR.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in Amazon EMR. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my Amazon EMR resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Amazon EMR supports these features, see How Amazon EMR works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.

Using Amazon S3 Access Grants with Amazon EMR

S3 Access Grants overview for Amazon EMR

With Amazon EMR releases 6.15.0 and higher, Amazon S3 Access Grants provide a scalable access control solution that you can use to augment access to your Amazon S3 data from Amazon EMR. If you have a complex or large permission configuration for your S3 data, you can use Access Grants to scale S3 data permissions for users, roles, and applications on your cluster.

Use S3 Access Grants to augment access to Amazon S3 data beyond the permissions that are granted by the runtime role or the IAM roles that are attached to the identities with access to your EMR cluster. For more information, see Managing access with S3 Access Grants in the *Amazon S3 User Guide*.

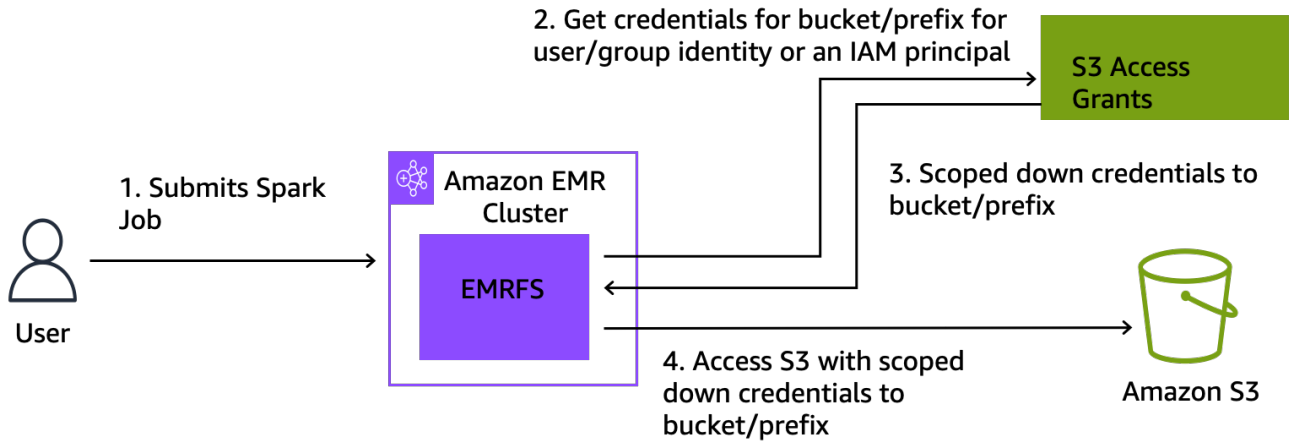
For steps to use S3 Access Grants with other Amazon EMR deployments, see the following documentation:

- Using S3 Access Grants with Amazon EMR on EKS
- Using S3 Access Grants with Amazon EMR Serverless

How Amazon EMR works with S3 Access Grants

Amazon EMR releases 6.15.0 and higher provide a native integration with S3 Access Grants. You can enable S3 Access Grants on Amazon EMR and run Spark jobs. When a Spark job makes a request for S3 data, Amazon S3 provides temporary credentials that are scoped to the specific bucket, prefix, or object.

The following is a high-level overview of how Amazon EMR gets access to data that's protected by S3 Access Grants.



1. A user submits an Amazon EMR Spark job that uses data stored in Amazon S3.
2. Amazon EMR makes a request for S3 Access Grants to allow access to the bucket, prefix, or object on behalf of that user.
3. Amazon S3 returns temporary credentials in the form of an AWS Security Token Service (STS) token for the user. The token is scoped to access the S3 bucket, prefix, or object.
4. Amazon EMR uses the STS token to retrieve data from S3.
5. Amazon EMR receives the data from S3 and returns the results to the user.

S3 Access Grants considerations with Amazon EMR

Take note of the following behaviors and limitations when you use S3 Access Grants with Amazon EMR.

Feature support

- S3 Access Grants is supported with Amazon EMR releases 6.15.0 and higher.
- Spark is the only supported query engine when you use S3 Access Grants with Amazon EMR.

- Delta Lake and Hudi are the only supported open-table formats when you use S3 Access Grants with Amazon EMR.
- The following Amazon EMR capabilities are not supported for use with S3 Access Grants:
  - Apache Iceberg tables
  - LDAP native authentication
  - Apache Ranger native authentication
  - AWS CLI requests to Amazon S3 that use IAM roles
  - S3 access through the open-source S3A protocol
- The `fallbackToIAM` option isn't supported for EMR clusters that use trusted identity propagation with IAM Identity Center.
- S3 Access Grants with AWS Lake Formation is only supported with Amazon EMR clusters that run on Amazon EC2.

Behavioral considerations

- The Apache Ranger native integration with Amazon EMR holds functionality that is congruent with S3 Access Grants as part of the EMRFS S3 Apache Ranger plugin. If you use Apache Ranger for fine-grained access control (FGAC), we recommend that you use that plugin instead of S3 Access Grants.
- Amazon EMR provides a credentials cache in EMRFS to ensure that a user doesn't need to make repeated requests for the same credentials within a Spark job. Therefore, Amazon EMR always requests the default-level privilege when it requests credentials. For more information, see Request access to S3 data in the *Amazon S3 User Guide*.
- In the case that a user performs an action that S3 Access Grants doesn't support, Amazon EMR is set to use the IAM role that was specified for job execution. For more information, see Fall back to IAM roles.

Launch an Amazon EMR cluster with S3 Access Grants

This section describes how to launch an EMR cluster that runs on Amazon EC2, and uses S3 Access Grants to manage access to data in Amazon S3. For steps to use S3 Access Grants with other Amazon EMR deployments, see the following documentation:

- Using S3 Access Grants with Amazon EMR on EKS

- Using S3 Access Grants with EMR Serverless

Use the following steps to launch an EMR cluster that runs on Amazon EC2, and uses S3 Access Grants to manage access to data in Amazon S3.

1. Set up a job execution role for your EMR cluster. Include the required IAM permissions that you need to run Spark jobs, `s3:GetDataAccess` and `s3:GetAccessGrantsInstanceForPrefix`:

```
{
    "Effect": "Allow",
    "Action": [
    "s3:GetDataAccess",
    "s3:GetAccessGrantsInstanceForPrefix"
    ],
    "Resource": [     //LIST ALL INSTANCE ARNS THAT THE ROLE IS ALLOWED TO QUERY
         "arn:aws_partition:s3:Region:account-id1:access-grants/default",
         "arn:aws_partition:s3:Region:account-id2:access-grants/default"
    ]
}
```

> **Note**
>
> With Amazon EMR, S3 Access Grants augment the permissions that are set in IAM roles. If the IAM roles that you specify for job execution contain permissions to access S3 directly, then users might be able to access more data than just the data that you define in S3 Access Grants.

2. Next, use the AWS CLI to create a cluster with Amazon EMR 6.15 or higher and the `emrfs-site` classification to enable S3 Access Grants, similar to the following example:

```
aws emr create-cluster
  --release-label emr-6.15.0 \
  --instance-count 3 \
  --instance-type m5.xlarge \
  --configurations '[{"Classification":"emrfs-site",
 "Properties":{"fs.s3.s3AccessGrants.enabled":"true",
 "fs.s3.s3AccessGrants.fallbackToIAM":"false"}}]'
```

S3 Access Grants with AWS Lake Formation

If you use Amazon EMR with the AWS Lake Formation integration, you can use Amazon S3 Access Grants for direct or tabular access to data in Amazon S3.

> **Note**
>
> S3 Access Grants with AWS Lake Formation is only supported with Amazon EMR clusters that run on Amazon EC2.

Direct access

Direct access involves all calls to access S3 data that don't invoke the API for the AWS Glue service that Lake Formation uses as a metastore with Amazon EMR, for example, to call `spark.read`:

```
spark.read.csv("s3://...")
```

When you use S3 Access Grants with AWS Lake Formation on Amazon EMR, all direct access patterns go through S3 Access Grants to get temporary S3 credentials.

Tabular access

Tabular access occurs when Lake Formation invokes the metastore API to access your S3 location, for example, to query table data:

```
spark.sql("select * from test_tbl")
```

When you use S3 Access Grants with AWS Lake Formation on Amazon EMR, all tabular access patterns go through Lake Formation.

Fall back to IAM roles

If a user attempts to perform an action that S3 Access Grants doesn't support, Amazon EMR defaults to the IAM role that was specified for job execution when the `fallbackToIAM` configuration is `true`. This allows users to fall back on their job execution role to give credentials for S3 access in scenarios that S3 Access Grants doesn't cover.

With `fallbackToIAM` enabled, users can access the data that the Access Grant allows. If there isn't an S3 Access Grants token for the target data, then Amazon EMR checks for the permission on their job execution role.

> **Note**
>
> We recommend that you test your access permissions with the `fallbackToIAM` configuration enabled even if you plan to disable the option for production workloads. With Spark jobs, there are other ways that users might be able to access all permission sets with their IAM credentials. When enabled on EMR clusters, grants from S3 give Spark jobs access to S3 locations. You should ensure that you protect these S3 locations from access outside of EMRFS. For example, you should protect the S3 locations from access by S3 clients used in notebooks, or by applications that aren't supported by S3 Access Grants such as Hive or Presto.

Authenticate to Amazon EMR cluster nodes

SSH clients can use an Amazon EC2 key pair to authenticate to cluster instances. Alternatively, with Amazon EMR releases 5.10.0 and higher, you can configure Kerberos to authenticate users and SSH connections to the primary node. And with Amazon EMR releases 5.12.0 and higher, you can authenticate with LDAP.

Topics

- Use an EC2 key pair for SSH credentials for Amazon EMR
- Use Kerberos for authentication with Amazon EMR
- Use Active Directory or LDAP servers for authentication with Amazon EMR

Use an EC2 key pair for SSH credentials for Amazon EMR

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 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 more information about setting .pem file permissions and how to connect to an EMR cluster's primary 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 Amazon EMR cluster primary node using SSH.

Use Kerberos for authentication with Amazon EMR

Amazon EMR releases 5.10.0 and higher support Kerberos. Kerberos is a network authentication protocol that 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 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 with Amazon EMR
- Kerberos architecture options with Amazon EMR
- Configuring Kerberos on Amazon EMR
- Using SSH to connect to Kerberized clusters with Amazon EMR
- Tutorial: Configure an cluster-dedicated KDC with Amazon EMR
- Tutorial: Configure a cross-realm trust with an Active Directory domain

Supported applications with Amazon EMR

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 for different Amazon EMR releases.

The Livy user interface is the only web user interface hosted on the cluster that is Kerberized.

- **Hadoop MapReduce**

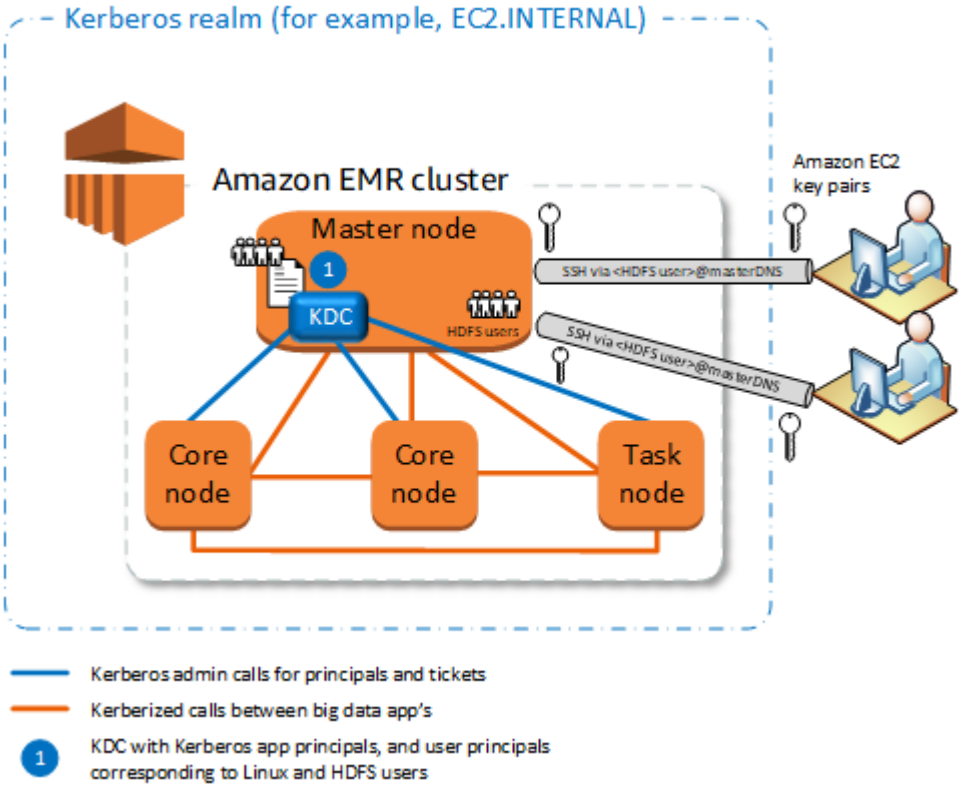
- **Hbase**
- **HCatalog**
- **HDFS**
- **Hive**
  - Do not enable Hive with LDAP authentication. This may cause issues communicating with Kerberized YARN.
- **Hue**
  - Hue user authentication isn't set automatically and can be configured using the configuration API.
  - Hue server is Kerberized. The 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 releases 5.22.0 and higher.
- **Oozie**
- **Phoenix**
- **Presto**
  - Presto supports Kerberos authentication in Amazon EMR releases 6.9.0 and higher.
  - To use Kerberos authentication for Presto, you must enable in-transit encryption.
- **Spark**
- **Tez**
- **Trino**
  - Trino supports Kerberos authentication in Amazon EMR releases 6.11.0 and higher.
  - To use Kerberos authentication for Trino, you must enable in-transit encryption.
- **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 with Amazon EMR

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.

Cluster-dedicated KDC (KDC on primary node)

This configuration is available with Amazon EMR releases 5.10.0 and higher.



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.

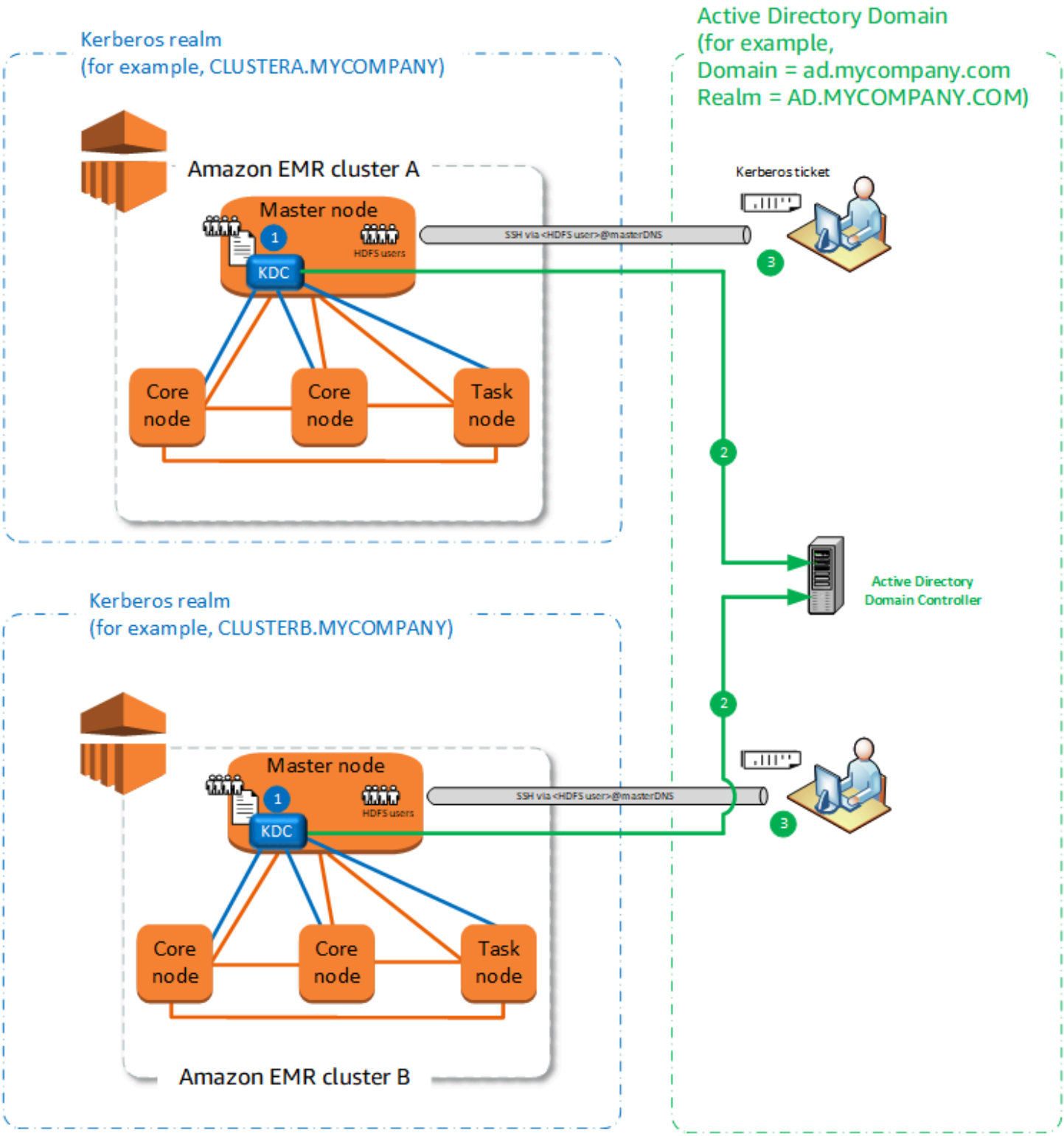
- Optionally, other Kerberized clusters can reference the KDC as an external KDC. For more information, see External KDC—primary node on a different cluster.

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 primary 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 primary 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.



Kerberos admin calls for principals and tickets

Kerberized calls between big data app's

1 KDC with Kerberos app principals and user principals

2 An incoming, one-way, cross-realm trust established in Active Directory

3 Users authenticate to the cluster using Kerberos credentials. An EC2 key pair is not required.

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.

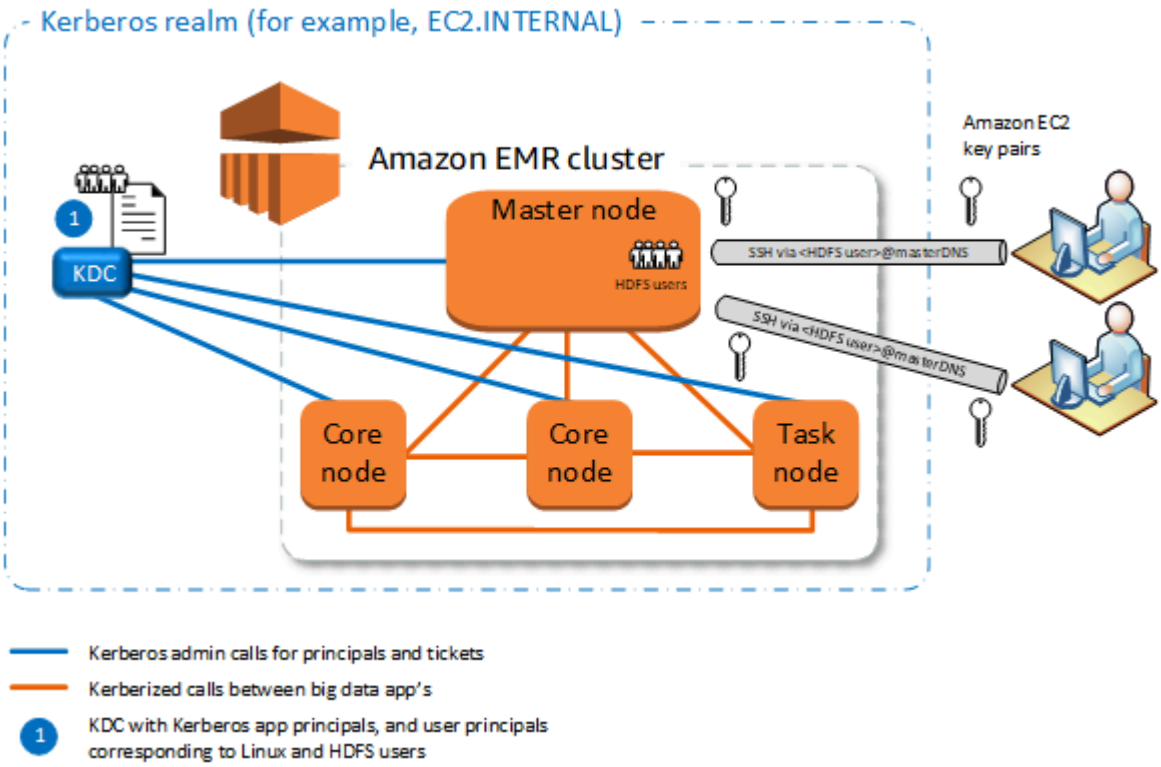
External KDC

Configurations with an External KDC are supported with Amazon EMR 5.20.0 and later.

- External KDC—MIT KDC
- External KDC—primary node on a different cluster
- External KDC—cluster KDC on a different cluster with Active Directory cross-realm trust

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. For more information, see Requirements for using multiple clusters with the same KDC.
- The primary node on a Kerberized cluster does not have the performance burden associated with maintaining the KDC.

Considerations and limitations

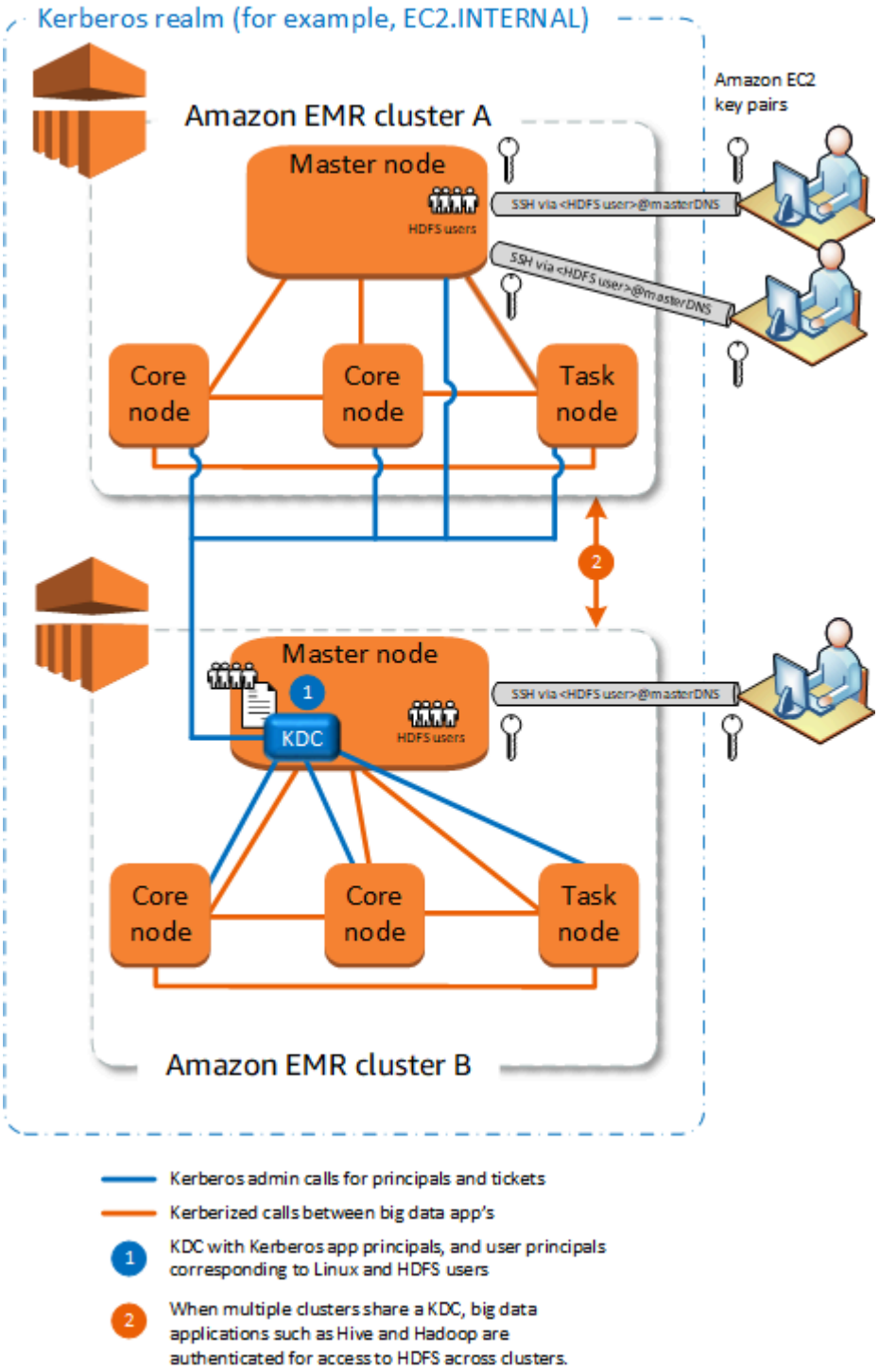
- You must create Linux users on the EC2 instance of each Kerberized cluster's primary node that correspond to KDC user principals, along with the HDFS directories for each user.
- User principals must use an EC2 private key file and `kinit` credentials to connect to Kerberized clusters using SSH.
- 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—primary node on a different cluster

This configuration is nearly identical to the external MIT KDC implementation above, except that the KDC is on the primary node of an EMR cluster. For more information, see Cluster-dedicated KDC (KDC on primary node) and Tutorial: Configure a cross-realm trust with an Active Directory domain.



Advantages

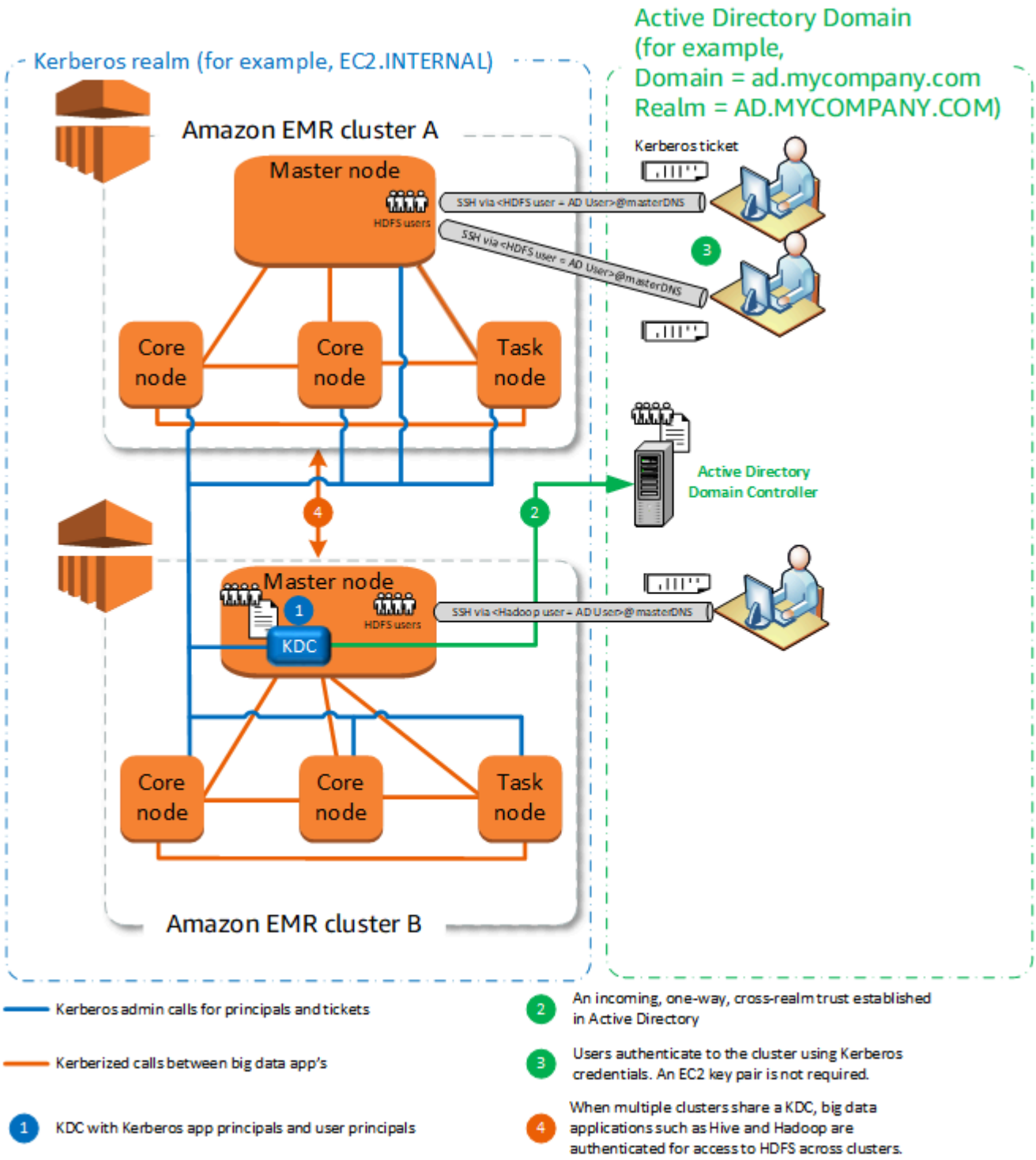
- Managing principals is consolidated in a single KDC.
- Multiple clusters can use the same KDC in the same Kerberos realm. For more information, see Requirements for using multiple clusters with the same KDC.

Considerations and limitations

- You must create Linux users on the EC2 instance of each Kerberized cluster's primary 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. 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. 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. For more information, see Requirements for using multiple clusters with the same KDC.
- 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 primary 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.

Requirements for using multiple clusters with the same KDC

Multiple clusters can use the same KDC in the same Kerberos realm. However, if the clusters concurrently run, then the clusters might fail if they use Kerberos ServicePrincipal names that conflict.

If you have multiple concurrent clusters with the same external KDC, then ensure that the clusters use different Kerberos realms. If the clusters must use the same Kerberos realm, then ensure that the clusters are in different subnets, and that their CIDR ranges don’t overlap.

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 Amazon EMR cluster security. For information about Kerberos properties in a security configuration, see Kerberos settings for security configurations.

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.

Step 3: Configure the cluster primary 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 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 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 an Amazon EMR cluster for Kerberos-authenticated HDFS users and SSH connections.

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 primary node, and the client SSH application

must be configured to use GSSAPI. For more information, see Configuring an Amazon EMR cluster for Kerberos-authenticated HDFS users and SSH connections.

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 with the Amazon EMR console or with the AWS CLI.

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 or Configuration examples.

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 primary 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

Parameter		Description
		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 configura tion 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 primary 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 configura tion. Additional configuration in the other Kerberos realm is required. For more information, see Tutorial: Configure a cross-realm trust with an Active Directory domain.

Parameter		Description
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.

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.

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 identity to join the cluster to the domain. For more information,

Parameter	Description
	see the section called "Step 3: Add accounts to the domain for the EMR Cluster".
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 accounts to the domain for the EMR Cluster".

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 primary node. Additional configuration on the cluster is required. For more information, see Configuring an Amazon EMR cluster for Kerberos-authenticated HDFS users and SSH connections.

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-7.10.0 \
--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 primary 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.

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-7.10.0 \
--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 primary node of a different cluster to authenticate principals. Additional configuration on the cluster is required. For more information, see Configuring an Amazon EMR cluster for Kerberos-authenticated HDFS users and SSH connections.

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-7.10.0 \
--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 primary node of another cluster to authenticate principals. That KDC has a cross-realm trust with an Active Directory domain controller. Additional configuration on the primary node with the KDC is required. For more information, see Tutorial: Configure a cross-realm trust with an Active Directory domain.

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-7.10.0 \
--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 an Amazon EMR 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 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 primary 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 primary node. In addition, users must have an SSH client with GSSAPI enabled.

Adding Linux users and Kerberos principals to the primary node

If you do not use Active Directory, you must create Linux accounts on the cluster primary node and add principals for these Linux users to the KDC. This includes a principal in the KDC for the primary node. In addition to the user principals, the KDC running on the primary 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 and External KDC—cluster KDC on a different cluster with Active Directory cross-realm trust.

> **Important**
>
> The KDC, along with the database of principals, is lost when the primary node terminates because the primary 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 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 primary node using an EC2 key pair as the default hadoop user to run the commands. For more information, see Connect to the Amazon EMR cluster primary node using SSH.

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://amzn-s3-demo-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 primary node, using the primary 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 primary 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 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 primary node using an EC2 key pair as the default `hadoop` user to run the commands. For more information, see Connect to the Amazon EMR cluster primary node using SSH.

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://amzn-
s3-demo-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 primary node using SSH, the SSH service must have GSSAPI authentication enabled. To enable GSSAPI, run the following commands from the primary 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 with Amazon EMR

This section demonstrates the steps for a Kerberos-authenticated user to connect to the primary 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 primary 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 an Amazon EMR cluster.

SSH uses GSSAPI for authenticating Kerberos clients, and you must enable GSSAPI authentication for the SSH service on the cluster primary node. For more information, see Enabling GSSAPI for SSH. 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 primary node.

To copy the krb5.conf file

1. Use SSH to connect to the primary node using an EC2 key pair and the default `hadoop` user —for example, `hadoop@MasterPublicDNS`. For detailed instructions, see Connect to an Amazon EMR cluster.
2. From the primary node, copy the contents of the `/etc/krb5.conf` file . For more information, see Connect to an Amazon EMR cluster.
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.

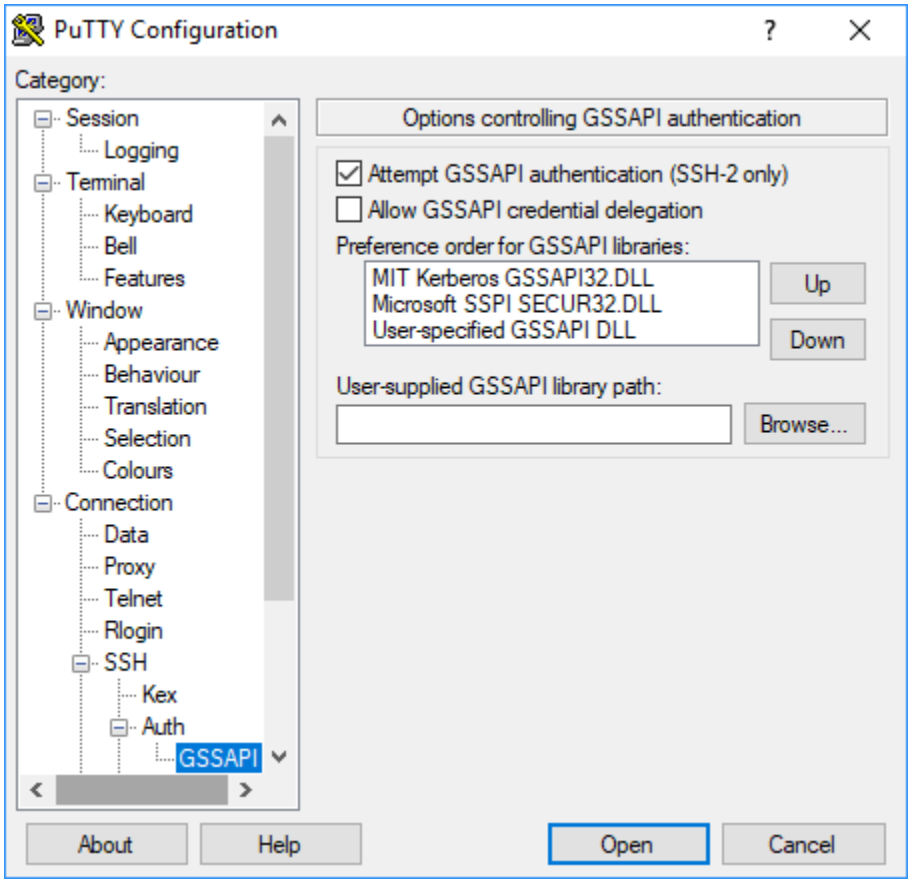
   Example: Linux users

   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 an cluster-dedicated KDC with Amazon EMR

This topic guides you through creating a cluster with a cluster-dedicated *key distribution center (KDC)*, manually adding Linux accounts to all cluster nodes, adding Kerberos principals to the KDC on the primary 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 for authentication with Amazon EMR.

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-7.10.0 \
--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://amzn-s3-demo-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 primary 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 primary 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://amzn-s3-
demo-bucket/configurekdc.sh"]
```

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

```
#!/bin/bash
#Add a principal to the KDC for the primary node, using the primary 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 primary 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 with Amazon EMR.

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.

For more information on Amazon EMR support for Kerberos and KDC, as well as links to MIT Kerberos Documentation, see Use Kerberos for authentication with Amazon EMR.

> **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

Step 2: Launch and install the Active Directory domain controller

Step 3: Add accounts to the domain for the EMR Cluster

Step 4: Configure an incoming trust on the Active Directory domain controller

Step 5: Use a DHCP option set to specify the Active Directory domain controller as a VPC DNS server

Step 6: Launch a Kerberized EMR Cluster

Step 7: Create HDFS users and set permissions on the cluster for Active Directory accounts

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.

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 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 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*.
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. 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*.
3. Connect to the EC2 instance using RDP. For more information, see Connecting to your Windows instance in the *Amazon EC2 User Guide*.

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 accounts to the domain for the EMR Cluster

RDP to the Active Directory domain controller to create accounts in Active Directory Users and Computers for each cluster user. For more information, see Create a User Account in Active Directory Users and Computers on the *Microsoft Learn* site. Make a note of each user's **User logon name**. You need these later when you configure the cluster.

In addition, create a 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. To delegate computer join privileges to the 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 primary 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.

```
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. 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 is associated with the cluster primary 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.
- 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 accounts to the domain for the EMR Cluster.

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 accounts

When setting up a trust relationship with Active Directory, Amazon EMR creates Linux users on the cluster for each Active Directory account. For example, the user logon name `LiJuan` in Active Directory has a Linux 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 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 primary node. Use the EC2 key pair that you specified when you created the cluster to connect to the primary node over SSH as the Hadoop user. For more information, see Use an EC2 key pair for SSH credentials for Amazon EMR.

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://amzn-s3-demo-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 primary node as described in Using SSH to connect to Kerberized clusters with Amazon EMR, 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.

Use Active Directory or LDAP servers for authentication with Amazon EMR

With Amazon EMR releases 6.12.0 and higher, you can use the LDAP over SSL (LDAPS) protocol to launch a cluster that natively integrates with your corporate identity server. LDAP (Lightweight Directory Access Protocol) is an open, vendor-neutral application protocol that accesses and maintains data. LDAP is commonly used for user authentication against corporate identity servers that are hosted on applications such as Active Directory (AD) and OpenLDAP. With this native integration, you can use your LDAP server to authenticate users on Amazon EMR.

Highlights of the Amazon EMR LDAP integration include:

- Amazon EMR configures the supported applications to authenticate with LDAP authentication on your behalf.
- Amazon EMR configures and maintains security for the supported applications with the Kerberos protocol. You don't need to input any commands or scripts.
- You get fine-grained access control (FGAC) through Apache Ranger authorization for Hive Metastore database and tables. See Integrate Amazon EMR with Apache Ranger for more information.
- When you require LDAP credentials to access a cluster, you get fine-grained access control (FGAC) over who can access your EMR clusters through SSH.

The following pages provide a conceptual overview, prerequisites, and steps to launch an EMR cluster with the Amazon EMR LDAP integration.

Topics

- Overview of LDAP with Amazon EMR
- LDAP components for Amazon EMR
- Application support and considerations with LDAP for Amazon EMR
- Configure and launch an EMR cluster with LDAP
- Examples using LDAP with Amazon EMR

Overview of LDAP with Amazon EMR

Lightweight Directory Access Protocol (LDAP) is a software protocol that network administrators use to manage and control access to data by authenticating users within a company's network. The LDAP protocol stores information in a hierarchical, tree directory structure. For more information, see Basic LDAP Concepts on *LDAP.com*.

Within a company's network, many applications might use the LDAP protocol to authenticate users. With the Amazon EMR LDAP integration, EMR clusters can natively use the same LDAP protocol with an added security configuration.

There are two major implementations of the LDAP protocol that Amazon EMR supports: **Active Directory** and **OpenLDAP**. While other implementations are possible, most fit the same authentication protocols as Active Directory or OpenLDAP.

Active Directory (AD)

Active Directory (AD) is a directory service from Microsoft for Windows domain networks. AD is included on most Windows Server operating systems, and can communicate with clients over the LDAP and LDAPS protocols. For authentication, Amazon EMR attempts a user-bind with your AD instance with the User Principal Name (UPN) as the distinguished name and password. The UPN uses the standard format `username@domain_name`.

OpenLDAP

OpenLDAP is a free, open-source implementation of the LDAP protocol. For authentication, Amazon EMR attempts a user-bind with your OpenLDAP instance with the fully qualified domain name (FQDN) as the distinguished name and password. The FQDN uses the standard format `username_attribute=username,LDAP_user_search_base`. Commonly, the `username_attribute` value is `uid`, and the `LDAP_user_search_base` value contains the attributes of the tree that leads to the user. For example, `ou=People,dc=example,dc=com`.

Other free and open-source implementations of the LDAP protocol typically follow a similar FQDN as OpenLDAP for the distinguished names of their users.

LDAP components for Amazon EMR

You can use your LDAP server to authenticate with Amazon EMR and any applications that the user directly utilizes on the EMR cluster through the following components.

Secret Agent

The *Secret Agent* is an on-cluster process that authenticates all user requests. The Secret Agent creates the user bind to your LDAP server on behalf of the supported applications on the EMR cluster. The Secret Agent runs as the `emrsecretagent` user, and it writes logs to the `/emr/secretagent/log` directory. These logs provide details about the state of each user's authentication request and any errors that might surface during user authentication.

System Security Services Daemon (SSSD)

SSSD is a daemon that runs on each node of an LDAP-enabled EMR cluster. SSSD creates and manages a UNIX user to sync your remote corporate identity to each node. YARN-based applications such as Hive and Spark require that a local UNIX user exists on every node that runs a query for a user.

Application support and considerations with LDAP for Amazon EMR

This topic lists supported applications, supported features and unsupported features.

Supported applications with LDAP for Amazon EMR

> **Important**
>
> The applications listed on this page are the only applications that Amazon EMR supports for LDAP. To ensure cluster security, you can only include LDAP-compatible applications when you create an EMR cluster with LDAP enabled. If you attempt to install other, unsupported applications, Amazon EMR will reject your request for a new cluster.

The Amazon EMR releases 6.12 and higher support LDAP integration with the following applications:

- Apache Livy
- Apache Hive through HiveServer2 (HS2)
- Trino
- Presto
- Hue

You can also install the following applications on an EMR cluster and configure them to meet your security needs:

- Apache Spark
- Apache Hadoop

Supported features with LDAP for Amazon EMR

You can use the following Amazon EMR features with the LDAP integration:

> **Note**
>
> To keep LDAP credentials secure, you must use in-transit encryption to secure the flow of data on and off the cluster. For more information about in-transit encryption, see Encrypt data at rest and in transit with Amazon EMR.

- Encryption in transit (required) and at rest
- Instance groups, instance fleets, and Spot Instances
- Reconfiguration of applications on a running cluster
- EMRFS server-side encryption (SSE)

Unsupported features

Consider the following limitations when you use the Amazon EMR LDAP integration:

- Amazon EMR disables steps for clusters with LDAP enabled.
- Amazon EMR doesn't support runtime roles and AWS Lake Formation integrations for clusters with LDAP enabled.

- Amazon EMR doesn't support LDAP with StartTLS.
- Amazon EMR doesn't support high-availability mode (clusters with multiple primary nodes) for clusters with LDAP enabled.
- You can't rotate bind credentials or certificates for clusters with LDAP enabled. If any of those fields were rotated, we recommend that you start a new cluster with the updated bind credentials or certificates.
- You must use exact search bases with LDAP. The LDAP user and group search base doesn't support LDAP search filters.

Configure and launch an EMR cluster with LDAP

This section covers how to configure Amazon EMR for use with LDAP authentication.

Topics

- Add AWS Secrets Manager permissions to the Amazon EMR instance role
- Create the Amazon EMR security configuration for LDAP integration
- Launch an EMR cluster that authenticates with LDAP

Add AWS Secrets Manager permissions to the Amazon EMR instance role

Amazon EMR uses an IAM service role 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 that Amazon EMR assigns to every EC2 instance in a cluster at launch.

To define permissions for an EMR cluster to interact with Amazon S3 data and other AWS services, define a custom Amazon EC2 instance profile 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 with Amazon EMR.

Add the following statements to the default EC2 instance profile to allow Amazon EMR to tag sessions and access the AWS Secrets Manager that stores LDAP certificates.

```
    {
      "Sid": "AllowAssumeOfRolesAndTagging",
      "Effect": "Allow",
```

```
      "Action": ["sts:TagSession", "sts:AssumeRole"],
      "Resource": [
        "arn:aws:iam::111122223333:role/LDAP_DATA_ACCESS_ROLE_NAME",
        "arn:aws:iam::111122223333:role/LDAP_USER_ACCESS_ROLE_NAME"
      ]
    },
    {
        "Sid": "AllowSecretsRetrieval",
        "Effect": "Allow",
        "Action": "secretsmanager:GetSecretValue",
        "Resource": [
            "arn:aws:secretsmanager:us-east-1:111122223333:secret:LDAP_SECRET_NAME*",
            "arn:aws:secretsmanager:us-
east-1:111122223333:secret:ADMIN_LDAP_SECRET_NAME*"
        ]
    }
```

> **Note**
>
> Your cluster requests will fail if you forget the wildcard * character at the end of the secret name when you set Secrets Manager permissions. The wildcard represents the secret versions.
>
> You should aslo limit the scope of the AWS Secrets Manager policy to only the certificates that your cluster needs to provision instances.

Create the Amazon EMR security configuration for LDAP integration

Before you can launch an EMR cluster with LDAP integration, use the steps in Create a security configuration with the Amazon EMR console or with the AWS CLI to create an Amazon EMR security configuration for the cluster. Complete the following configurations in the `LDAPConfiguration` block under `AuthenticationConfiguration`, or the in corresponding fields in the Amazon EMR console **Security Configurations** section:

`EnableLDAPAuthentication`

Console option: **Authentication protocol: LDAP**

To use the LDAP integration, set this option to `true` or select it as your authentication protocol when you create a cluster in the console. By default, `EnableLDAPAuthentication` is `true` when you create a security configuration in the Amazon EMR console.

`LDAPServerURL`

Console option: **LDAP server location**

The location of the LDAP server including the prefix: `ldaps://`*`location_of_server`*.

`BindCertificateARN`

Console option: **LDAP SSL certificate**

The AWS Secrets Manager ARN that contains the certificate to sign the SSL certificate that the LDAP server uses. If your LDAP server is signed by a public Certificate Authority (CA), you can provide an AWS Secrets Manager ARN with a blank file. For more information on how to store your certificate in Secrets Manager, see Store TLS certificates in AWS Secrets Manager.

`BindCredentialsARN`

Console option: **LDAP server bind credentials**

An AWS Secrets Manager ARN that contains the LDAP admin user bind credentials. The credentials are stored as a JSON object. There is only one key-value pair in this secret; the key in the pair is the username, and the value is the password. For example, `{"uid=admin,cn=People,dc=example,dc=com": "AdminPassword1"}`. This is an optional field unless you enable SSH login for your EMR cluster. In many configurations, Active Directory instances require bind credentials to allow SSSD to sync users.

`LDAPAccessFilter`

Console option: **LDAP access filter**

Specifies the subset of objects within your LDAP server that can authenticate. For example, if all you want to grant access to all users with the `posixAccount` object class in your LDAP server, define the access filter as `(objectClass=posixAccount)`.

`LDAPUserSearchBase`

Console option: **LDAP user search base**

The search base that your users belong under within your LDAP server. For example, `cn=People,dc=example,dc=com`.

`LDAPGroupSearchBase`

Console option: **LDAP group search base**

The search base that your groups belong under within your LDAP server. For example, `cn=Groups,dc=example,dc=com`.

`EnableSSHLogin`

Console option: **SSH login**

Specifies whether or not to allow password authentication with LDAP credentials. We don't recommend that you enable this option. Key pairs are a more secure route to allow access into EMR clusters. This field is optional and defaults to `false`.

`LDAPServerType`

Console option: **LDAP server type**

Specifies the type of LDAP server that Amazon EMR connects to. Supported options are Active Directory and OpenLDAP. Other LDAP server types might work, but Amazon EMR doesn't officially support other server types. For more information, see LDAP components for Amazon EMR.

`ActiveDirectoryConfigurations`

A required sub-block for security configurations that use the Active Directory server type.

`ADDomain`

Console option: **Active Directory domain**

The domain name used to create the User Principal Name (UPN) for user authentication with security configurations that use the Active Directory server type.

Considerations for security configurations with LDAP and Amazon EMR

- To create a security configuration with Amazon EMR LDAP integration, you must use in-transit encryption. For information about in-transit encryption, see Encrypt data at rest and in transit with Amazon EMR.
- You can't define Kerberos configuration in the same security configuration. Amazon EMR provisions a KDC thar is dedicated to the automatically, and manages the admin password for this KDC. Users can't access this admin password.
- You can't define IAM runtime roles and AWS Lake Formation in the same security configuration.
- The `LDAPServerURL` must have the `ldaps://` protocol in its value.
- The `LDAPAccessFilter` can't be empty.

Use LDAP with the Apache Ranger integration for Amazon EMR

With the LDAP integration for Amazon EMR, you can further integrate with Apache Ranger. When you pull .your LDAP users into Ranger, you can then associate those users with an Apache Ranger policy server to integrate with Amazon EMR and other applications. To do this, define the `RangerConfiguration` field within `AuthorizationConfiguration` in the security configuration that you use with your LDAP cluster. For more information on how to set up the security configuration, see Create the EMR security configuration.

When you use LDAP with Amazon EMR, you don't need to provide a `KerberosConfiguration` with the Amazon EMR integration for Apache Ranger.

Launch an EMR cluster that authenticates with LDAP

Use the following steps to launch an EMR cluster with LDAP or Active Directory.

1. Set up your environment:
   - Make sure that the nodes on your EMR cluster can communicate with Amazon S3 and AWS Secrets Manager. For more information on how to modify your EC2 instance profile role to communicate with these services, see Add AWS Secrets Manager permissions to the Amazon EMR instance role.
   - If you plan to run your EMR cluster in a private subnet, you should use AWS PrivateLink and Amazon VPC endpoints, or use network address transalation (NAT) to configure the VPC to communicate with S3 and Secrets Manager. For more information, see AWS PrivateLink and VPC endpoints and NAT instances in the *Amazon VPC Getting Started Guide*.
   - Make sure that there is network connectivity between your EMR cluster and the LDAP server. Your EMR clusters must access your LDAP server over the network. The primary, core, and task nodes for the cluster communicate with the LDAP server to sync user data. If your LDAP server runs on Amazon EC2, update the EC2 security group to accept traffic from the EMR cluster. For more information, see Add AWS Secrets Manager permissions to the Amazon EMR instance role.
2. Create an Amazon EMR security configuration for the LDAP integration. For more information, see Create the Amazon EMR security configuration for LDAP integration.
3. Now that you're set up, use the steps in Launch an Amazon EMR cluster to launch your cluster with the following configurations:
   - Select Amazon EMR release 6.12 or higher. We recommend that you use the latest Amazon EMR release.

- Only specify or select applications for your cluster that support LDAP. For a list of LDAP-supported applications with Amazon EMR, see Application support and considerations with LDAP for Amazon EMR.
- Apply the security configuration that you created in the previous step.

Examples using LDAP with Amazon EMR

Once you provision an EMR cluster that uses LDAP integration, you can provide your LDAP credentials to any supported application through its built-in username and password authentication mechanism. This page shows some examples.

Using LDAP authentication with Apache Hive

Example - Apache Hive

The following example command starts an Apache Hive session through HiveServer2 and Beeline:

```
beeline -u "jdbc:hive2:///$HOSTNAME:10000/default;ssl=true;sslTrustStore=
$TRUSTSTORE_PATH;trustStorePassword=$TRUSTSTORE_PASS"  -n LDAP_USERNAME -
p LDAP_PASSWORD
```

Using LDAP authentication with Apache Livy

Example - Apache Livy

The following example command starts a Livy session through cURL. Replace *ENCODED-KEYPAIR* with a Base64-encoded string for `username:password`.

```
curl -X POST --data '{"proxyUser":"LDAP_USERNAME","kind": "pyspark"}' -H "Content-Type:
 application/json" -H "Authorization: Basic ENCODED-KEYPAIR" DNS_OF_PRIMARY_NODE:8998/
sessions
```

Using LDAP authentication with Presto

Example - Presto

The following example command starts a Presto session through the Presto CLI:

```
presto-cli --user "LDAP_USERNAME" --password --catalog hive
```

After you run this command, enter the LDAP password at the prompt.

Using LDAP authentication with Trino

Example - Trino

The following example command starts a Trino session through the Trino CLI:

```
trino-cli --user "LDAP_USERNAME" --password --catalog hive
```

After you run this command, enter the LDAP password at the prompt.

Using LDAP authentication with Hue

You can access Hue UI through an SSH tunnel that you create on the cluster, or you can set a proxy server to publicly broadcast the connection to Hue. Because Hue doesn't run in HTTPS mode by default, we recommend that you use an additional encryption layer to ensure that communication between clients and the Hue UI is encrypted with HTTPS. This reduces the chance that you might accidentally expose user credentials in plain text.

To use the Hue UI, open the Hue UI in your browser and enter your LDAP username password to log in. If the credentials are correct, Hue logs you in and uses your identity to authenticate you with all supported applications.

Using SSH for password authentication and Kerberos tickets for other applications

> **Important**
>
> We don't recommend that you use password authentication to SSH into an EMR cluster.

You can use your LDAP credentials to SSH to an EMR cluster. To do this, set the `EnableSSHLogin` configuration to `true` in the Amazon EMR security configuration that you use to start the cluster. Then, use the following command to SSH to the cluster once its been launched:

```
ssh username@EMR_PRIMARY_DNS_NAME
```

After you run this command, enter the LDAP password at the prompt.

Amazon EMR includes an on-cluster script that allows users to generate a Kerberos keytab file and ticket to use with supported applications that don't accept LDAP credentials directly. Some of these applications include `spark-submit`, Spark SQL, and PySpark.

Run `ldap-kinit` and follow the prompts. If the authentication succeeds, the Kerberos keytab file appears in your home directory with a valid Kerberos ticket. Use the Kerberos ticket to run applications as you would on any Kerberized environment.

Integrate Amazon EMR with AWS IAM Identity Center

With Amazon EMR releases 6.15.0 and higher, you can use identities from AWS IAM Identity Center to authenticate with an Amazon EMR cluster. The following sections provides a conceptual overview, prerequisites, and steps required to launch an EMR cluster with Identity Center integration.

Topics

- Overview
- Features and benefits
- Getting started with AWS IAM Identity Center integration for Amazon EMR
- Considerations and limitations for Amazon EMR with the Identity Center integration

Overview

Identity Center is the recommended approach for workforce authentication and authorization on AWS for organizations of any size and type. With Identity Center, you can create and manage user identities in AWS, or connect your existing identity source, including Microsoft Active Directory, Okta, Ping Identity, JumpCloud, Google Workspace, and Microsoft Entra ID (formerly Azure AD).

Trusted identity propagation is an AWS IAM Identity Center feature that administrators of connected AWS services can use to grant and audit access to service data. Access to this data is based on user attributes such as group associations. Setting up trusted identity propagation requires collaboration between the administrators of connected AWS services and the IAM Identity Center administrators. For more information, see Prerequisites and considerations.

Features and benefits

The Amazon EMR integration with IAM Identity Center provides the following benefits:

- Amazon EMR provides credentials to relay your Identity Center Identity to an EMR cluster.
- Amazon EMR configures all supported applications to authenticate with the cluster credentials.

- Amazon EMR configures and maintains the supported application security with the Kerberos protocol and no commands or scripts required by you.
- The ability to enforce Amazon S3 prefix-level authorization with Identity Center identities on S3 Access Grants-managed S3 prefixes.
- The ability to enforce table-level authorization with Identity Center identities on AWS Lake Formation managed AWS Glue tables.

Getting started with AWS IAM Identity Center integration for Amazon EMR

This section helps you configure Amazon EMR to integrate with AWS IAM Identity Center.

Topics

- Create an Identity Center instance
- Create an IAM role for Identity Center
- Add permissions for services not integrated with IAM Identity Center
- Create an Identity Center enabled security configuration
- Create and launch an Identity Center enabled cluster
- Configure Lake Formation for an IAM Identity Center enabled EMR cluster
- Working with S3 Access Grants on an IAM Identity Center enabled EMR cluster

> **Note**
>
> In order to use Identity Center integration with EMR, Lake Formation or S3 Access Grants must be enabled. You can also use both. If neither is enabled, Identity Center integration isn't supported.

Create an Identity Center instance

If you don't already have one, create an Identity Center instance in the AWS Region where you want to launch your EMR cluster. An Identity Center instance can only exist in a single Region for an AWS account.

Use the following AWS CLI command to create a new instance named *MyInstance*:

```
aws sso-admin create-instance --name MyInstance
```

Create an IAM role for Identity Center

To integrate Amazon EMR with AWS IAM Identity Center, create an IAM role that authenticates with Identity Center from the EMR cluster. Under the hood, Amazon EMR uses SigV4 credentials to relay the Identity Center identity to downstream services such as AWS Lake Formation. Your role should also have the respective permissions to invoke the downstream services.

When you create the role, use the following permissions policy:

```
{
  "Statement": [
    {
      "Sid": "IdCPermissions",
      "Effect": "Allow",
      "Action": [
        "sso-oauth:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "GlueandLakePermissions",
      "Effect": "Allow",
      "Action": [
        "glue:*",
        "lakeformation:GetDataAccess"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AccessGrantsPermissions",
      "Effect": "Allow",
      "Action": [
        "s3:GetDataAccess",
        "s3:GetAccessGrantsInstanceForPrefix"
      ],
      "Resource": "*"
    }
  ]
}
```

The trust policy for this role allows the InstanceProfile role to let it assume the role.

```
{
    "Sid": "AssumeRole",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::12345678912:role/EMR_EC2_DefaultRole"
    },
    "Action": [
        "sts:AssumeRole",
        "sts:SetContext"
    ]
}
```

If the role doesn't have trusted credentials and accesses a Lake Formation-protected table, Amazon EMR automatically sets the `principalId` of the assumed role to *`userID`*`-untrusted`. The following is a snippet of a CloudTrail event that displays the `principalId`.

```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "ABCDEFGH1JKLMNO2PQR3TU:5000-untrusted",
        "arn": "arn:aws:sts::123456789012:assumed-role/EMR_TIP/5000-untrusted",
        "accountId": "123456789012",
        "accessKeyId": "ABCDEFGH1IJKLMNOPQ7R3"
        ...
```

Add permissions for services not integrated with IAM Identity Center

AWS credentials that use Trusted Identity Propagation the IAM policies defined in the IAM role for any calls made to services not integrated with IAM Identity Center. This includes, for example, the AWS Key Management Service. Your role should also define any IAM permissions for any such services you would attempt to access. Currently supported IAM Identity Center integrated services include AWS Lake Formation and Amazon S3 Access Grants.

To learn more about Trusted Identity Propagation, see Trusted Identity Propagation across applications.

Create an Identity Center enabled security configuration

To launch an EMR cluster with IAM Identity Center integration, use the following example command to create an Amazon EMR security configuration that has Identity Center enabled. Each configuration is explained below.

```
aws emr create-security-configuration --name "IdentityCenterConfiguration-with-lf-
accessgrants" --region "us-west-2" --security-configuration '{
    "AuthenticationConfiguration":{
        "IdentityCenterConfiguration":{
            "EnableIdentityCenter":true,
            "IdentityCenterApplicationAssigmentRequired":false,
            "IdentityCenterInstanceARN": "arn:aws:sso:::instance/
ssoins-123xxxxxxxxx789"
        }
    },
    "AuthorizationConfiguration": {
        "LakeFormationConfiguration": {
            "AuthorizedSessionTagValue": "Amazon EMR"
        },
        "IAMConfiguration": {
          "EnableApplicationScopedIAMRole": true,
          "ApplicationScopedIAMRoleConfiguration": {
            "PropagateSourceIdentity": true
          }
        }
    },
    "EncryptionConfiguration": {
        "EnableInTransitEncryption": true,
        "EnableAtRestEncryption": false,
        "InTransitEncryptionConfiguration": {
            "TLSCertificateConfiguration": {
                "CertificateProviderType": "PEM",
                "S3Object": "s3://amzn-s3-demo-bucket/cert/my-certs.zip"
            }
        }
    }
}'
```

- **`EnableIdentityCenter`** – (required) Enables Identity Center integration.

- **`IdentityCenterInstanceARN`** – (optional) The Identity Center instance ARN. If this isn't included, the existing IAM Identity Center instance ARN is looked up as part of the configuration step.
- **`IAMRoleForEMRIdentityCenterApplicationARN`** – (required) The IAM role that procures Identity Center tokens from the cluster.
- **`IdentityCenterApplicationAssignmentRequired`** – (boolean) Governs if an assignment will be required to use the Identity Center application. This field is optional. If a value isn't provided, the default is `false`.
- **`AuthorizationConfiguration` / `LakeFormationConfiguration`** – Optionally, configure authorization:
  - **`IAMConfiguration`** – Enables EMR Runtimes Roles feature to be used in addition to your TIP identity. If you enable this configuration, then you (or the caller AWS Service) will be required to specify an IAM Runtime Role in each call to the EMR Steps or EMR `GetClusterSessionCredentials` APIs. If the EMR cluster is being used with SageMaker Unified Studio, then this option is required if Trusted Identity Propagation is also enabled.
  - **`EnableLakeFormation`** – Enable Lake Formation authorization on the cluster.

To enable Identity Center integration with Amazon EMR, you must specify `EncryptionConfiguration` and `IntransitEncryptionConfiguration`.

Create and launch an Identity Center enabled cluster

Now that you've set up the IAM role that authenticates with Identity Center, and created an Amazon EMR security configuration that has Identity Center enabled, you can create and launch your identity-aware cluster. For steps to launch your cluster with the required security configuration, see Specify a security configuration for an Amazon EMR cluster.

The following sections describe how to configure your Identity Center enabled cluster with security options that Amazon EMR supports:

- Working with S3 Access Grants on an IAM Identity Center enabled EMR cluster
- Configure Lake Formation for an IAM Identity Center enabled EMR cluster

Configure Lake Formation for an IAM Identity Center enabled EMR cluster

You can integrate AWS Lake Formation with your AWS IAM Identity Center enabled EMR cluster.

First, be sure you have an Identity Center instance set up in the same Region as your cluster. For more information, see Create an Identity Center instance. You can find the instance ARN in the IAM Identity Center console when you view the instance details, or use the following command to view details for all your instances from the CLI:

```
aws sso-admin list-instances
```

Then use the ARN and your AWS account ID with the following command to configure Lake Formation to be compatible with IAM Identity Center:

```
aws lakeformation create-lake-formation-identity-center-configuration --cli-input-json
 file://create-lake-fromation-idc-config.json
json input:
{
    "CatalogId": "account-id/org-account-id",
    "InstanceArn": "identity-center-instance-arn"
}
```

Now, call `put-data-lake-settings` and enable `AllowFullTableExternalDataAccess` with Lake Formation:

```
aws lakeformation put-data-lake-settings --cli-input-json file://put-data-lake-
settings.json
json input:
{
    "DataLakeSettings": {
        "DataLakeAdmins": [
            {
                "DataLakePrincipalIdentifier": "admin-ARN"
            }
        ],
        "CreateDatabaseDefaultPermissions": [...],
        "CreateTableDefaultPermissions": [...],
        "AllowExternalDataFiltering": true,
        "AllowFullTableExternalDataAccess": true
    }
}
```

Finally, grant full table permissions to the identity ARN for the user that accesses the EMR cluster. The ARN contains the user ID from Identity Center. Navigate to Identity Center in the console, select **Users**, and then select the user to view their **General information** settings.

Copy the User ID and paste it into the following ARN for *user-id*:

```
arn:aws:identitystore:::user/user-id
```

> **Note**
>
> Queries on the EMR cluster only work if the IAM Identity Center identity has full table access on the Lake Formation protected table. If the identity doesn't have full table access, then the query will fail.

Use the following command to grant the user full table access:

```
aws lakeformation grant-permissions --cli-input-json file://grantpermissions.json
json input:
{
    "Principal": {
        "DataLakePrincipalIdentifier": "arn:aws:identitystore:::user/user-id"
    },
    "Resource": {
        "Table": {
            "DatabaseName": "tip_db",
            "Name": "tip_table"
        }
    },
    "Permissions": [
        "ALL"
    ],
    "PermissionsWithGrantOption": [
        "ALL"
    ]
}
```

Adding the application ARN to IDC for Lake Formation integration

In order to query Lake Formation enabled resources, the Application ARN of the IDC application needs to be added. To do this, follow these steps:

1. On the console, choose **AWS Lake Formation**.
2. Select **IAM Identity Center integration** and **Lake Formation application integration** by matching the application ARN. The ARN will appear in the **Application ID** list.

Working with S3 Access Grants on an IAM Identity Center enabled EMR cluster

You can integrate S3 Access Grants with your AWS IAM Identity Center enabled EMR cluster.

Use S3 Access Grants to authorize access to your data sets from clusters that use Identity Center. Create grants to augment the permissions that you set for IAM users, groups, roles, or for a corporate directory. For more information, see Using S3 Access Grants with Amazon EMR.

Topics

- Create an S3 Access Grants instance and location
- Create grants for Identity Center identities

Create an S3 Access Grants instance and location

If you don't already have one, create an S3 Access Grants instance in the AWS Region where you want to launch your EMR cluster.

Use the following AWS CLI command to create a new instance named *MyInstance*:

```
aws s3control-access-grants create-access-grants-instance \
--account-id 12345678912 \
--identity-center-arn "identity-center-instance-arn" \
```

Then, create an S3 Access Grants location, replacing the red values with your own:

```
aws s3control-access-grants create-access-grants-location \
--account-id 12345678912 \
--location-scope s3:// \
--iam-role-arn "access-grant-role-arn" \
--region aa-example-1
```

> **Note**
>
> Define the `iam-role-arn` parameter as the `accessGrantRole` ARN.

Create grants for Identity Center identities

Finally, create the grants for the identities that have access to your cluster:

```
aws s3control-access-grants create-access-grant \
--account-id 12345678912 \
--access-grants-location-id "default" \
--access-grants-location-configuration S3SubPrefix="s3-bucket-prefix"
--permission READ \
--grantee GranteeType=DIRECTORY_USER,GranteeIdentifier="your-identity-center-user-id"
```

Example Output:

```
{
"CreatedAt": "2023-09-21T23:47:24.870000+00:00",
"AccessGrantId": "1234-12345-1234-1234567",
"AccessGrantArn": "arn:aws:s3:aa-example-1-1:123456789012:access-grants/default/grant/
xxxx1234-1234-5678-1234-1234567890",
"Grantee": {
"GranteeType": "DIRECTORY_USER",
"GranteeIdentifier": "5678-56789-5678-567890"
},
"AccessGrantsLocationId": "default",
"AccessGrantsLocationConfiguration": {
"S3SubPrefix": "myprefix/*"
},
"Permission": "READ",
"GrantScope": "s3://myprefix/*"
}
```

Considerations and limitations for Amazon EMR with the Identity Center integration

Consider the following points when you use IAM Identity Center with Amazon EMR:

- Trusted Identity Propagation through Identity Center is supported on Amazon EMR 6.15.0 and higher, and only with Apache Spark. Also, Trusted Identity Propagation through Identity Center using EMR Runtime Roles feature is supported on Amazon EMR 7.8.0 and higher, and only with Apache Spark.
- To enable EMR clusters with trusted identity propagation, you must use the AWS CLI to create a security configuration that has trusted identity propagation enabled, and use that security configuration when you launch your cluster. For more information, see Create an Identity Center enabled security configuration.

- Fine-grained access controls using AWS Lake Formation that use Trusted Identity Propagation are available for Amazon EMR clusters on EMR version 7.2.0 and higher. Between EMR versions 6.15.0 and 7.1.0, only table-level access control, based on AWS Lake Formation, is available.
- With Amazon EMR clusters that use Trusted Identity Propagation, operations that support access control based on Lake Formation with Apache Spark include SELECT, ALTER TABLE, INSERT INTO, and DROP TABLE.
- Trusted Identity Propagation with Amazon EMR is supported in the following AWS Regions:
  - `af-south-1` – Africa (Cape Town)
  - `ap-east-1` – Asia Pacific (Hong Kong)
  - `ap-northeast-1` – Asia Pacific (Tokyo)
  - `ap-northeast-2` – Asia Pacific (Seoul)
  - `ap-northeast-3` – Asia Pacific (Osaka)
  - `ap-south-1` – Asia Pacific (Mumbai)
  - `ap-south-2` – Asia Pacific (Hyderabad)
  - `ap-southeast-1` – Asia Pacific (Singapore)
  - `ap-southeast-2` – Asia Pacific (Sydney)
  - `ap-southeast-3` – Asia Pacific (Jakarta)
  - `ap-southeast-4` – Asia Pacific (Melbourne)
  - `ca-central-1` – Canada (Central)
  - `eu-central-1` – Europe (Frankfurt)
  - `eu-central-2` – Europe (Zurich)
  - `eu-north-1` – Europe (Stockholm)
  - `eu-south-1` – Europe (Milan)
  - `eu-south-2` – Europe (Spain)
  - `eu-west-1` – Europe (Ireland)
  - `eu-west-2` – Europe (London)
  - `eu-west-3` – Europe (Paris)
  - `il-central-1` – Israel (Tel Aviv)
  - `me-central-1` – Middle East (UAE)
  - `me-south-1` – Middle East (Bahrain)

  - `sa-east-1` – South America (São Paulo)

- `us-east-1` – US East (N. Virginia)
- `us-east-2` – US East (Ohio)
- `us-west-1` – US West (N. California)
- `us-west-2` – US West (Oregon)

Integrate Amazon EMR with AWS Lake Formation

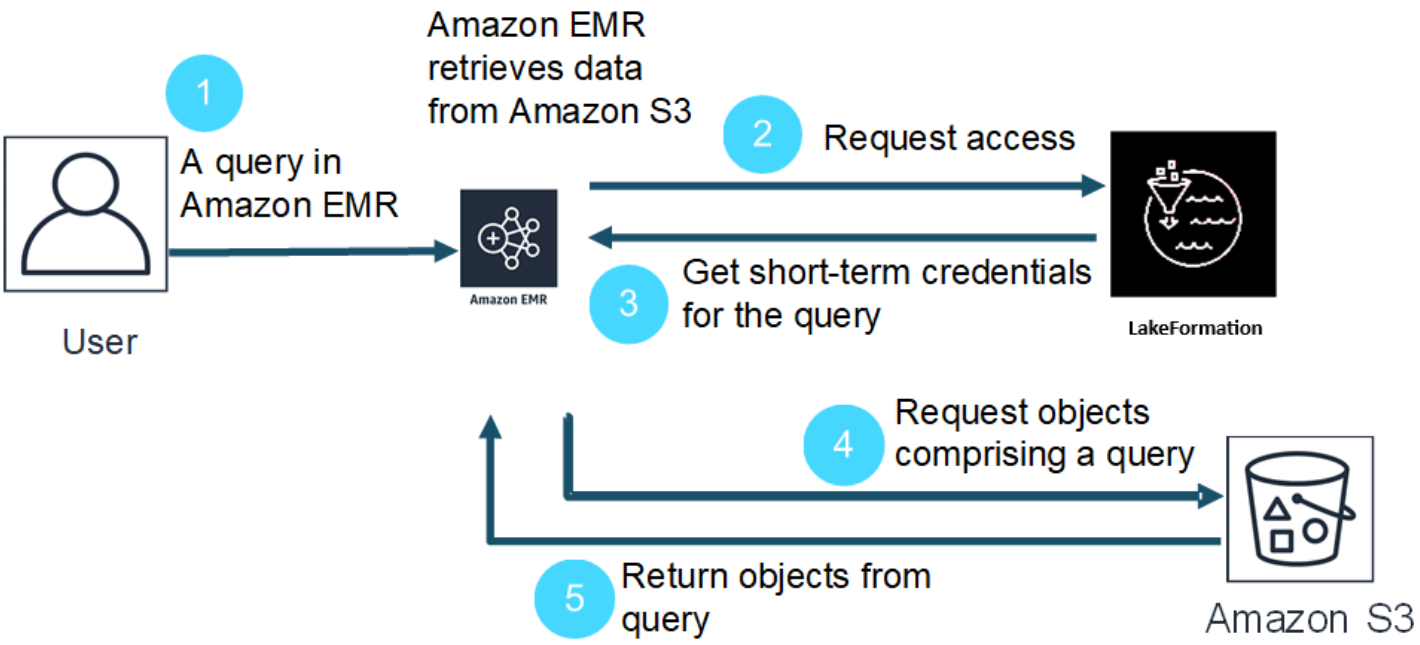
AWS Lake Formation is a managed service that helps you discover, catalog, cleanse, and secure data in an Amazon Simple Storage Service (S3) data lake. Lake Formation provides fine-grained, column-level access to databases and tables in the AWS Glue Data Catalog. For more information, see What is AWS Lake Formation?

With Amazon EMR release 6.7.0 and later, you can apply Lake Formation based access control to Spark, Hive, and Presto jobs that you submit to Amazon EMR clusters. To integrate with Lake Formation, you must create an EMR cluster with a *runtime role*. A runtime role is an AWS Identity and Access Management (IAM) role that you associate with Amazon EMR jobs or queries. Amazon EMR then uses this role to access AWS resources. For more information, see Runtime roles for Amazon EMR steps.

How Amazon EMR works with Lake Formation

After you integrate Amazon EMR with Lake Formation, you can execute queries to Amazon EMR clusters with the `Step` API or with SageMaker AI Studio. Then, Lake Formation provides access to data through temporary credentials for Amazon EMR. This process is called credential vending. For more information, see What is AWS Lake Formation?

The following is a high-level overview of how Amazon EMR gets access to data protected by Lake Formation security policies.



1. A user submits an Amazon EMR query for data in Lake Formation.
2. Amazon EMR requests temporary credentials from Lake Formation to give the user data access.
3. Lake Formation returns temporary credentials.
4. Amazon EMR sends the query request to retrieve data from Amazon S3.
5. Amazon EMR receives the data from Amazon S3, filters it, and returns results based on the user permissions that the user defined in Lake Formation.

For more information about adding users and groups to Lake Formation policies, see Granting Data Catalog permissions.

Prerequisites

You must meet the following requirements before you integrate Amazon EMR and Lake Formation:

- Turn on runtime role authorization on your Amazon EMR cluster.
- Use the AWS Glue Data Catalog as your metadata store.
- Define and manage permissions in Lake Formation to access databases, tables, and columns in AWS Glue Data Catalog. For more information, see What is AWS Lake Formation?

Topics

- Enable Lake Formation with Amazon EMR
- Apache Hudi and Lake Formation with Amazon EMR
- Apache Iceberg and Lake Formation with Amazon EMR
- Delta Lake and Lake Formation on Amazon EMR
- Considerations for Amazon EMR with Lake Formation
- Working with Glue Data Catalog views in Amazon EMR
- Using a Lake Formation job runtime role with full table access

Enable Lake Formation with Amazon EMR

With Amazon EMR 6.15.0 and higher, when you run Spark jobs on Amazon EMR on EC2 clusters that access data in the AWS Glue Data Catalog, you can use AWS Lake Formation to apply table, row, column, and cell level permissions on Hudi, Iceberg, or Delta Lake based tables.

In this section, we cover how to create a security configuration and set up Lake Formation to work with Amazon EMR. We also go over how to launch a cluster with the security configuration that you created for Lake Formation.

Step 1: Set up a runtime role for your EMR cluster

To use a runtime role for your EMR cluster, you must create a security configuration. With a security configuration, you can apply consistent security, authorization, and authentication options across your clusters.

1. Create a file called `lf-runtime-roles-sec-cfg.json` with the following security configuration.

```
{
    "AuthorizationConfiguration": {
        "IAMConfiguration": {
            "EnableApplicationScopedIAMRole": true,
            "ApplicationScopedIAMRoleConfiguration": {
                "PropagateSourceIdentity": true
            }
        },
        "LakeFormationConfiguration": {
            "AuthorizedSessionTagValue": "Amazon EMR"
        }
    },
```

```
    "EncryptionConfiguration": {
     "EnableAtRestEncryption": false,
            "EnableInTransitEncryption": true,
            "InTransitEncryptionConfiguration": {
            "TLSCertificateConfiguration": {<certificate-configuration>}
        }
    }
}
```

The example below illustrates how to use a zip file with certificates in Amazon S3 for certificate configuration:

- 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 for certificate requirements.)

```
"TLSCertificateConfiguration": {
 "CertificateProviderType": "PEM",
 "S3Object": "s3://MyConfigStore/artifacts/MyCerts.zip"
 }
```

The example below illustrates how to use a custom key provider for certificate configuration:

- A custom key provider is used. (See Providing certificates for encrypting data in transit with Amazon EMR encryption for certificate requirements.)

```
"TLSCertificateConfiguration": {
 "CertificateProviderType": "Custom",
 "S3Object": "s3://MyConfig/artifacts/MyCerts.jar",
 "CertificateProviderClass": "com.mycompany.MyCertProvider"
                         }
```

2. Next, to ensure that the session tag can authorize Lake Formation, set the `LakeFormationConfiguration/AuthorizedSessionTagValue` property to `Amazon EMR`.
3. Use the following command to create the Amazon EMR security configuration.

```
aws emr create-security-configuration \
--name 'iamconfig-with-iam-lf' \
```

```
--security-configuration file://lf-runtime-roles-sec-cfg.json
```

Alternatively, you can use the Amazon EMR console to create a security configuration with custom settings.

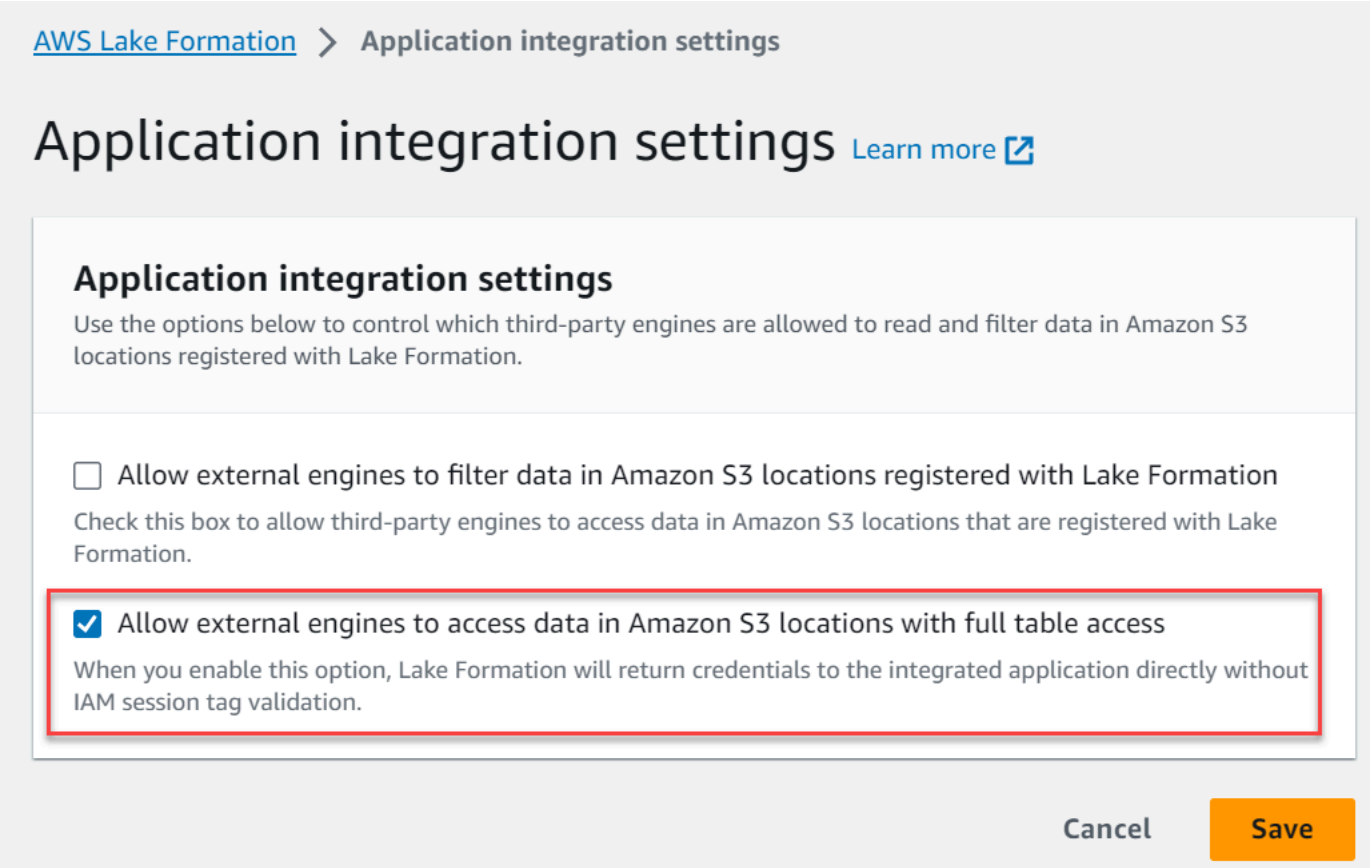
Step 2: Launch an Amazon EMR cluster

Now you're ready to launch an EMR cluster with the security configuration that you created in the previous step. For more information on security configurations, see Use security configurations to set up Amazon EMR cluster security and Runtime roles for Amazon EMR steps.

Step 3a: Set up Lake Formation-based table-level permissions with Amazon EMR runtime roles

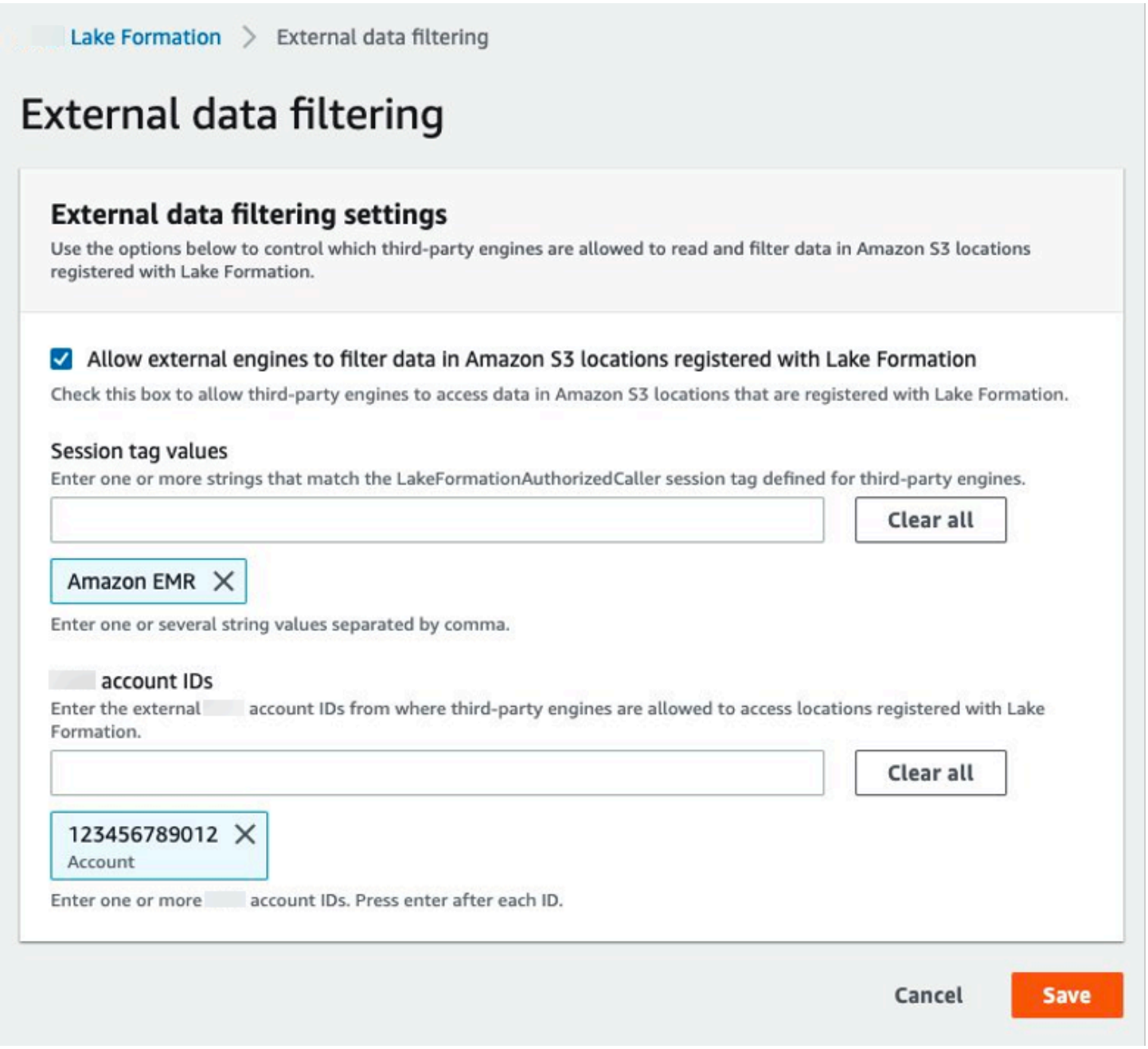
If you don't require fine-grained access control at the column, row, or cell level, you can set up table-level permissions with Glue Data Catalog. To enable table-level access, navigate to the AWS Lake Formation console and select the **Application integration settings** option from the **Administration** section in the sidebar. Then, enable the following option and choose **Save**:

Allow external engines to access data in Amazon S3 locations with full table access



Step 3b: Set up Lake Formation-based column, row, or cell-level permissions with Amazon EMR runtime roles

To apply table and column level permissions with Lake Formation, the data lake administrator for Lake Formation must set `Amazon EMR` as the value for the session tag configuration, `AuthorizedSessionTagValue`. Lake Formation uses this session tag to authorize callers and provide access to the data lake. You can set this session tag in the **External data filtering** section of the Lake Formation console. Replace *123456789012* with your own AWS account ID.



Step 4: Configure AWS Glue and Lake Formation grants for Amazon EMR runtime roles

To continue with your setup of Lake Formation based access control with Amazon EMR runtime roles, you must configure AWS Glue and Lake Formation grants for Amazon EMR runtime roles. To allow your IAM runtime roles to interact with Lake Formation, grant them access with `lakeformation:GetDataAccess` and `glue:Get*`.

Lake Formation permissions control access to AWS Glue Data Catalog resources, Amazon S3 locations, and the underlying data at those locations. IAM permissions control access to the Lake Formation and AWS Glue APIs and resources. Although you might have the Lake Formation permission to access a table in the data catalog (SELECT), your operation fails if you don't have the IAM permission on the `glue:Get*` API. For more details about Lake Formation access control, see Lake Formation access control overview.

1. Create the `emr-runtime-roles-lake-formation-policy.json` file with the following content.

   JSON

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "LakeFormationManagedAccess",
         "Effect": "Allow",
         "Action": [
           "lakeformation:GetDataAccess",
           "glue:Get*",
           "glue:Create*",
           "glue:Update*"
         ],
         "Resource": [
           "*"
         ]
       }
     ]
   }
   ```

2. Create the related IAM policy.

   ```
   aws iam create-policy \
   --policy-name emr-runtime-roles-lake-formation-policy \
   --policy-document file://emr-runtime-roles-lake-formation-policy.json
   ```

3. To assign this policy to your IAM runtime roles, follow the steps in Managing AWS Lake Formation permissions.

You can now use runtime roles and Lake Formation to apply table and column level permissions. You can also use a source identity to control actions and monitor operations with AWS CloudTrail.

For each IAM role that you plan to use as a runtime role, set the following trust policy, replacing `EMR_EC2_DefaultRole` with your instance profile role. To modify the trust policy of an IAM role, 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",
        "sts:TagSession"
       ]
 }
```

For a detailed, end-to-end example, see Introducing runtime roles for Amazon EMR steps.

For information about how to integrate with Iceberg and AWS Glue Data Catalog for a multi-catalog hierarchy, see Configure Spark to access a multi-catalog hierarchy in AWS Glue Data Catalog.

Apache Hudi and Lake Formation with Amazon EMR

Amazon EMR releases 6.15.0 and higher include support for fine-grained access control based on AWS Lake Formation with Apache Hudi when you read and write data with Spark SQL. Amazon EMR supports table, row, column, and cell-level access control with Apache Hudi. With this feature, you can run snapshot queries on copy-on-write tables to query the latest snapshot of the table at a given commit or compaction instant.

Currently, a Lake Formation-enabled Amazon EMR cluster must retrieve Hudi's commit time column to perform incremental queries and time travel queries. It doesn't support Spark's `timestamp as of` syntax and the `Spark.read()` function. The correct syntax is `select * from table where _hoodie_commit_time <= point_in_time`. For more information, see Point in time Time-Travel queries on Hudi table.

The following support matrix lists some core features of Apache Hudi with Lake Formation:

	Copy on Write	Merge on Read
Snapshot queries - Spark SQL	✓	✓
Read-optimized queries - Spark SQL	✓	✓
Incremental queries	✓	✓
Time travel queries	✓	✓
Metadata tables	✓	✓
DML INSERT commands	✓	✓
DDL commands		
Spark datasource queries		
Spark datasource writes		

Querying Hudi tables

This section shows how you can run the supported queries described above on a Lake Formation enabled cluster. The table should be a registered catalog table.

1. To start the Spark shell, use the following commands.

```
spark-sql
--jars /usr/lib/hudi/hudi-spark-bundle.jar \
--conf spark.serializer=org.apache.spark.serializer.KryoSerializer \
--conf
spark.sql.catalog.spark_catalog=org.apache.spark.sql.hudi.catalog.HoodieCatalog \
--conf
spark.sql.extensions=org.apache.spark.sql.hudi.HoodieSparkSessionExtension,com.amazonaws.em
 \
--conf spark.sql.catalog.spark_catalog.lf.managed=true
```

   If you want Lake Formation to use record server to manage your Spark catalog, set `spark.sql.catalog.<managed_catalog_name>.lf.managed` to true.

2. To query the latest snapshot of copy-on-write tables, use the following commands.

```
SELECT * FROM my_hudi_cow_table
```

```
spark.read.table("my_hudi_cow_table")
```

3. To query the latest compacted data of MOR tables, you can query the read-optimized table that is suffixed with _ro:

```
SELECT * FROM my_hudi_mor_table_ro
```

```
spark.read.table("my_hudi_mor_table_ro")
```

> **Note**
>
> The performance of reads on Lake Formation clusters might be slower because of optimizations that are not supported. These features include file listing based on Hudi metadata, and data skipping. We recommend that you test your application performance to ensure that it meets your requirements.

Apache Iceberg and Lake Formation with Amazon EMR

Amazon EMR releases 6.15.0 and higher include support for fine-grained access control based on AWS Lake Formation with Apache Iceberg when you read and write data with Spark SQL. Amazon EMR supports table, row, column, and cell-level access control with Apache Iceberg. With this feature, you can run snapshot queries on copy-on-write tables to query the latest snapshot of the table at a given commit or compaction instant.

If you want to use Iceberg format, set the following configurations. Replace *DB_LOCATION* with the Amazon S3 path where your Iceberg tables are located, and replace the Region and account ID placeholders with your own values.

```
spark-sql \
--conf
 spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions,com.ama

--conf spark.sql.catalog.iceberg_catalog=org.apache.iceberg.spark.SparkCatalog
```

```
--conf spark.sql.catalog.iceberg_catalog.warehouse=s3://DB_LOCATION
--conf spark.sql.catalog.iceberg_catalog.catalog-
impl=org.apache.iceberg.aws.glue.GlueCatalog
--conf spark.sql.catalog.iceberg_catalog.io-impl=org.apache.iceberg.aws.s3.S3FileIO
--conf spark.sql.catalog.iceberg_catalog.glue.account-id=ACCOUNT_ID
--conf spark.sql.catalog.iceberg_catalog.glue.id=ACCOUNT_ID
--conf spark.sql.catalog.iceberg_catalog.client.assume-role.region=AWS_REGION
--conf spark.sql.secureCatalog=iceberg_catalog
```

If you want Lake Formation to use record server to manage your Spark catalog, set `spark.sql.catalog.<managed_catalog_name>.lf.managed` to true.

You should also **be careful NOT to pass** the following assume role settings:

```
--conf spark.sql.catalog.my_catalog.client.assume-role.region
--conf spark.sql.catalog.my_catalog.client.assume-role.arn
--conf spark.sql.catalog.my_catalog.client.assume-
role.tags.LakeFormationAuthorizedCaller
```

The following support matrix lists some core features of Apache Iceberg with Lake Formation:

	Copy on Write	Merge on Read
Snapshot queries - Spark SQL	✓	✓
Read-optimized queries - Spark SQL	✓	✓
Incremental queries	✓	✓
Time travel queries	✓	✓
Metadata tables	✓	✓
DML INSERT commands	✓	✓
DDL commands		
Spark datasource queries		
Spark datasource writes		

Delta Lake and Lake Formation on Amazon EMR

Amazon EMR releases 6.15.0 and higher include support for fine-grained access control based on AWS Lake Formation with Delta Lake when you read and write data with Spark SQL. Amazon EMR supports table, row, column, and cell-level access control with Delta Lake. With this feature, you can run snapshot queries on copy-on-write tables to query the latest snapshot of the table at a given commit or compaction instant.

To use Delta Lake with Lake Formation, run the following command.

```
spark-sql \
--conf
 spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension,com.amazonaws.emr.recordserver.co
 \
--conf spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog
 \
--conf spark.sql.catalog.spark_catalog.lf.managed=true
```

If you want Lake Formation to use record server to manage your Spark catalog, set `spark.sql.catalog.<managed_catalog_name>.lf.managed` to true.

The following support matrix lists some core features of Delta Lake with Lake Formation:

	Copy on Write	Merge on Read
Snapshot queries - Spark SQL	✓	✓
Read-optimized queries - Spark SQL	✓	✓
Incremental queries	Not supported	Not supported
Time travel queries	Not supported	Not supported
Metadata tables	✓	✓
DML `INSERT` commands	✓	✓
DDL commands		
Spark datasource queries		

	Copy on Write	Merge on Read
Spark datasource writes		

Creating a Delta Lake table in AWS Glue Data Catalog

Amazon EMR with Lake Formation doesn't support DDL commands and Delta table creation. Follow these steps to create tables in the AWS Glue Data Catalog.

1. Use the following example to create a Delta table. Make sure that your S3 location exists.

```
spark-sql \
--conf "spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension" \
--conf
 "spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog"

> CREATE DATABASE if not exists <DATABASE_NAME> LOCATION 's3://<S3_LOCATION>/
transactionaldata/native-delta/<DATABASE_NAME>/';
> CREATE TABLE <TABLE_NAME> (x INT, y STRING, z STRING) USING delta;
> INSERT INTO <TABLE_NAME> VALUES (1, 'a1', 'b1');
```

2. To see the details of your table, go to https://console.aws.amazon.com/glue/.
3. In the left navigation, expand **Data Catalog**, choose **Tables**, then choose the table you created. Under **Schema**, you should see that the Delta table you created with Spark stores all columns in a data type of `array<string>` in AWS Glue.
4. To define column and cell-level filters in Lake Formation, remove the `col` column from your schema, and then add the columns that are in your table schema. In this example, add the columns `x`, `y`, and `z`.

Considerations for Amazon EMR with Lake Formation

Consider the following when using Amazon EMR with AWS Lake Formation.

Amazon EMR with Lake Formation is available in all available regions.

- Amazon EMR supports fine-grained access control via Lake Formation only for Apache Hive and Apache Iceberg tables. Apache Hive formats include Parquet, ORC, and xSV.
- You can't turn off `DynamicResourceAllocation` for Lake Formation jobs.

- You can only use Lake Formation with Spark jobs.
- Amazon EMR with Lake Formation only supports a single Spark session throughout a job.
- Amazon EMR with Lake Formation only supports cross-account table queries shared through resource links.
- The following aren't supported:
  - Resilient distributed datasets (RDD)
  - Spark streaming
  - Write with Lake Formation granted permissions
  - Access control for nested columns
- Amazon EMR blocks functionalities that might undermine the complete isolation of system driver, including the following:
  - UDTs, HiveUDFs, and any user-defined function that involves custom classes
  - Custom data sources
  - Supply of additional jars for Spark extension, connector, or metastore
  - `ANALYZE TABLE` command
- To enforce access controls, `EXPLAIN PLAN` and DDL operations such as `DESCRIBE TABLE` don't expose restricted information.
- Amazon EMR restricts access to system driver Spark logs on Lake Formation-enabled applications. Since the system driver runs with elevated permissions, events and logs that the system driver generates can include sensitive information. To prevent unauthorized users or code from accessing this sensitive data, Amazon EMR disables access to system driver logs.

  System profile logs are always persisted in managed storage – this is a mandatory setting that cannot be disabled. These logs are stored securely and encrypted using either a Customer Managed KMS key or an AWS Managed KMS key.

  If your Amazon EMR application is in a private subnet with VPC endpoints for Amazon S3 and you attach an endpoint policy to control access, before your jobs can send log data to AWS Managed Amazon S3, you must include the permissions detailed in Managed storage in your VPC policy to S3 gateway endpoint. For troubleshooting requests, contact AWS support.
- If you registered a table location with Lake Formation, the data access path goes through the Lake Formation stored credentials regardless of the IAM permission for the Amazon EMR job runtime role. If you misconfigure the role registered with table location, jobs submitted that use the role with S3 IAM permission to the table location will fail.

- Writing to a Lake Formation table uses IAM permission rather than Lake Formation granted permissions. If your job runtime role has the necessary S3 permissions, you can use it to run write operations.

The following are considerations and limitations when using Apache Iceberg:

- You can only use Apache Iceberg with session catalog and not arbitrarily named catalogs.
- Iceberg tables that are registered in Lake Formation only support the metadata tables `history`, `metadata_log_entries`, `snapshots`, `files`, `manifests`, and `refs`. Amazon EMR hides the columns that might have sensitive data, such as `partitions`, `path`, and `summaries`. This limitation doesn't apply to Iceberg tables that aren't registered in Lake Formation.
- Tables that you don't register in Lake Formation support all Iceberg stored procedures. The `register_table` and `migrate` procedures aren't supported for any tables.
- We recommend that you use Iceberg DataFrameWriterV2 instead of V1.
- EMR 7.10 provides a way to switch back to RecordServer if you would like to use features supported by RecordServer, but not yet supported by native FGAC, such as writeback to Lake Formation registered tables. To switch back, specify the following configurations when launching the cluster.

```
{
    "Classification": "spark-defaults",
    "Properties": {
      "spark.emr.lakeformation.legacy.enabled": "true"
    }
  },
  {
    "Classification": "yarn-site",
    "Properties": {
      "spark.emr.lakeformation.legacy.enabled": "true"
    }
  }
```

Working with Glue Data Catalog views in Amazon EMR

> **Note**
>
> Creating and managing AWS Glue Data Catalog views for use with EMR on EC2 is available with Amazon EMR release 7.10.0 and later.

You can create and manage views in the AWS Glue Data Catalog for use with EMR on EC2. These are known commonly as AWS Glue Data Catalog views. These views are useful because they support multiple SQL query engines, so you can access the same view across different AWS services, such as EMR on EC2, Amazon Athena, and Amazon Redshift.

By creating a view in the Data Catalog, you can use resource grants and tag-based access controls in AWS Lake Formation to grant access to it. Using this method of access control, you don't have to configure additional access to the tables you referenced when creating the view. This method of granting permissions is called definer semantics, and these views are called definer views. For more information about access control in Lake Formation, see Granting and revoking permissions on Data Catalog resources in the AWS Lake Formation Developer Guide.

Data Catalog views are useful for the following use cases:

- **Granular access control** – You can create a view that restricts data access based on the permissions the user needs. For example, you can use views in the Data Catalog to prevent employees who don’t work in the HR department from seeing personally identifiable information (PII).
- **Complete view definition** – By applying filters on your view in the Data Catalog, you make sure that data records available in a view in the Data Catalog are always complete.
- **Enhanced security** – The query definition used to create the view must be complete. This benefit means that views in the Data Catalog are less susceptible to SQL commands from malicious actors.
- **Simple sharing data** – Share data with other AWS accounts without moving data. For more information, see Cross-account data sharing in Lake Formation.

Creating a Data Catalog view

There are different ways to create a Data Catalog view. These include using the AWS CLI or Spark SQL. A few examples follow.

Using SQL

The following shows the syntax for creating a Data Catalog view. Note the `MULTI DIALECT` view type. This distinguishes the Data Catalog view from other views. The `SECURITY` predicate is specified as `DEFINER`. This indicates a Data Catalog view with `DEFINER` semantics.

```
CREATE [ OR REPLACE ] PROTECTED MULTI DIALECT VIEW [IF NOT EXISTS] view_name
[(column_name [COMMENT column_comment], ...) ]
[ COMMENT view_comment ]
[TBLPROPERTIES (property_name = property_value, ... )]
SECURITY DEFINER
AS query;
```

The following is a sample `CREATE` statement, following the syntax:

```
CREATE PROTECTED MULTI DIALECT VIEW catalog_view
SECURITY DEFINER
AS
SELECT order_date, sum(totalprice) AS price
FROM source_table
GROUP BY order_date
```

You can also create a view in dry-run mode, using SQL, to test view creation, without actually creating the resource. Using this option results in a "dry run" that validates the input and, if the validation succeeds, returns the JSON of the AWS Glue table object that will represent the view. In this case, The actual view isn't created.

```
CREATE [ OR REPLACE ] PROTECTED MULTI DIALECT VIEW view_name
SECURITY DEFINER
[ SHOW VIEW JSON ]
AS view-sql
```

Using the AWS CLI

> **Note**
>
> When you use the CLI command, the SQL used to create the view isn't parsed. This can result in a case where the view is created, but queries aren't successful. Be sure to test your SQL syntax prior to creating the view.

You use the following CLI command to create a view:

```
aws glue create-table --cli-input-json '{
  "DatabaseName": "database",
  "TableInput": {
    "Name": "view",
    "StorageDescriptor": {
      "Columns": [
        {
          "Name": "col1",
          "Type": "data-type"
        },
        ...
        {
          "Name": "col_n",
          "Type": "data-type"
        }
      ],
      "SerdeInfo": {}
    },
    "ViewDefinition": {
      "SubObjects": [
        "arn:aws:glue:aws-region:aws-account-id:table/database/referenced-table1",
        ...
        "arn:aws:glue:aws-region:aws-account-id:table/database/referenced-tableN",
       ],
      "IsProtected": true,
      "Representations": [
        {
          "Dialect": "SPARK",
          "DialectVersion": "1.0",
          "ViewOriginalText": "Spark-SQL",
          "ViewExpandedText": "Spark-SQL"
```

```
        }
      ]
    }
  }
}'
```

Supported view operations

The following command fragments show you various ways to work with Data Catalog views:

- **CREATE VIEW**

  Creates a data-catalog view. The following is a sample that shows creating a view from an existing table:

  ```
  CREATE PROTECTED MULTI DIALECT VIEW catalog_view
  SECURITY DEFINER AS SELECT * FROM my_catalog.my_database.source_table
  ```

- **ALTER VIEW**

  Available syntax:

  - `ALTER VIEW view_name [FORCE] ADD DIALECT AS query`
  - `ALTER VIEW view_name [FORCE] UPDATE DIALECT AS query`
  - `ALTER VIEW view_name DROP DIALECT`

  You can use the `FORCE ADD DIALECT` option to force update the schema and sub objects as per the new engine dialect. Note that doing this can result in query errors if you don't also use `FORCE` to update other engine dialects. The following shows a sample:

  ```
  ALTER VIEW catalog_view FORCE ADD DIALECT
  AS
  SELECT order_date, sum(totalprice) AS price
  FROM source_table
  GROUP BY orderdate;
  ```

  The following shows how to alter a view in order to update the dialect:

  ```
  ALTER VIEW catalog_view UPDATE DIALECT AS
  SELECT count(*) FROM my_catalog.my_database.source_table;
  ```

- **DESCRIBE VIEW**

  Available syntax for describing a view:

  - `SHOW COLUMNS {FROM|IN} view_name [{FROM|IN} database_name]` – If the user has the required AWS Glue and Lake Formation permissions to describe the view, they can list the columns. The following shows a couple sample commands for showing columns:

    ```
    SHOW COLUMNS FROM my_database.source_table;
    SHOW COLUMNS IN my_database.source_table;
    ```

  - `DESCRIBE view_name` – If the user has the required AWS Glue and Lake Formation permissions to describe the view, they can list the columns in the view along with its metadata.

- **DROP VIEW**

  Available syntax:

  - `DROP VIEW [ IF EXISTS ] view_name`

    The following sample shows a `DROP` statement that tests if a view exists prior to dropping it:

    ```
    DROP VIEW IF EXISTS catalog_view;
    ```

- **SHOW CREATE VIEW**

  - `SHOW CREATE VIEW view_name` – Shows the SQL statement that creates the specified view. The following is a sample that shows creating a data-catalog view:

    ```
    SHOW CREATE TABLE my_database.catalog_view;
    CREATE PROTECTED MULTI DIALECT VIEW my_catalog.my_database.catalog_view (
      net_profit,
      customer_id,
      item_id,
      sold_date)
    TBLPROPERTIES (
      'transient_lastDdlTime' = '1736267222')
    SECURITY DEFINER AS SELECT * FROM
    my_database.store_sales_partitioned_lf WHERE customer_id IN (SELECT customer_id
     from source_table limit 10)
    ```

- **SHOW VIEWS**

List all views in the catalog such asregular views, multi-dialect views (MDV), and MDV without Spark dialect. Available syntax is the following:

- `SHOW VIEWS [{ FROM | IN } database_name] [LIKE regex_pattern]`:

  The following shows a sample command to show views:

  ```
  SHOW VIEWS IN marketing_analytics LIKE 'catalog_view*';
  ```

For more information about creating and configuring data-catalog views, see Building AWS Glue Data Catalog views in the AWS Lake Formation Developer Guide.

Querying a Data Catalog view

After creating a Data Catalog view, you can query it using an Amazon EMR Spark job that has AWS Lake Formation fine-grained access control enabled. The job runtime role must have the Lake Formation `SELECT` permission on the Data Catalog view. You don't need to grant access to the underlying tables referenced in the view.

Once you have everything set up, you can query your view. For example, after creating an Amazon EMR application in EMR Studio, you can run the following query to access a view.

```
SELECT * from my_database.catalog_view LIMIT 10;
```

A helpful function is the `invoker_principal`. It returns the unique identifier of the EMRS job runtime role. This can be used to control the view output, based on the invoking principal. You can use this to add a condition in your view that refines query results, based on the calling role. The job runtime role must have permission to the `LakeFormation:GetDataLakePrincipal` IAM action to use this function.

```
select invoker_principal();
```

You can add this function to a `WHERE` clause, for instance, to refine query results.

Considerations and limitations

When you create Data Catalog views, the following apply:

- You can only create Data Catalog views with Amazon EMR 7.10 and above.

- The Data Catalog view definer must have `SELECT` access to the underlying base tables accessed by the view. Creating the Data Catalog view fails if a specific base table has any Lake Formation filters imposed on the definer role.
- Base tables must not have the `IAMAllowedPrincipals` data lake permission in Lake Formation. If present, the error *Multi Dialect views may only reference tables without IAMAllowedPrincipals permissions* occurs.
- The table's Amazon S3 location must be registered as a Lake Formation data lake location. If the table isn't registered, the error *Multi Dialect views may only reference Lake Formation managed tables* occurs. For information about how to register Amazon S3 locations in Lake Formation, see Registering an Amazon S3 location in the AWS Lake Formation Developer Guide.
- You can only create `PROTECTED` Data Catalog views. `UNPROTECTED` views aren't supported.
- You can't reference tables in another AWS account in a Data Catalog view definition. You also can't reference a table in the same account that's in a separate region.
- To share data across an account or region, the entire view must be be shared cross account and cross region, using Lake Formation resource links.
- User-defined functions (UDFs) aren't supported.
- You can use views based on Iceberg tables. The open-table formats Apache Hudi and Delta Lake are also supported.
- You can't reference other views in Data Catalog views.
- An AWS Glue Data Catalog view schema is always stored using lowercase. For example, if you use a DDL statement to create a Glue Data Catalog view with a column named `Castle`, the column created in the Glue Data Catalog will be made lowercase, to `castle`. If you then specify the column name in a DML query as `Castle` or `CASTLE`, EMR Spark will make the name lowercase for you in order to run the query. But the column heading displays using the casing that you specified in the query.

  If you want a query to fail in a case where a column name specified in the DML query does not match the column name in the Glue Data Catalog, you can set `spark.sql.caseSensitive=true`.

Using a Lake Formation job runtime role with full table access

The topics in this section cover how you can access Lake Formation protected Glue Data catalog tables from EMR Spark jobs or interactive sessions with full table access.

Lake Formation unfiltered access for EMR on EC2

With Amazon EMR releases 7.8.0 and higher, you can leverage AWS Lake Formation with Glue Data Catalog where the job runtime role has full table permissions. This capability allows you to read and write to tables that are protected by Lake Formation from your EMR Spark batch and interactive jobs.

Using Lake Formation with full table access

You can access AWS Lake Formation protected Glue Data catalog tables from EMR Spark jobs or interactive sessions where the job's runtime role has full table access. You do not have to enable AWS Lake Formation on the EMR cluster. When a Spark job is configured for Full Table Access (FTA), AWS Lake Formation credentials are used to read/write S3 data for AWS Lake Formation registered tables, while the job's runtime role credentials will be used to read/write tables not registered with AWS Lake Formation.

> **Note**
>
> Lake Formation fine-grained access control isn't required or supported for a job in order to run Full Table Access (FTA).

> **Note**
>
> Full table access does not require use of an EMR runtime role, and it works with both the runtime role and service role for cluster EC2 instances. For more information, see Service role for cluster EC2 instances (EC2 instance profile) or Runtime roles for Amazon EMR steps.

Step 1: Enable Full Table Access in Lake Formation

To use Full Table Access (FTA) mode, you need to allow third-party query engines to access data without the IAM session tag validation in AWS Lake Formation. To enable, follow the steps in Application integration for full table access.

> **Note**
>
> When accessing cross account tables, full-table access must be enabled in both the producer and consumer accounts. In the same manner, when accessing cross-region tables, this setting must be enabled in both producer and consumer regions.

Step 2: Setup IAM permissions for job runtime role

For read or write access to underlying data, in addition to Lake Formation permissions, a job runtime role needs the `lakeformation:GetDataAccess` IAM permission. With this permission, Lake Formation grants the request for temporary credentials to access the data.

The following is an example policy of how to provide IAM permissions to access a script in Amazon S3, uploading logs to S3, AWS Glue API permissions, and permission to access Lake Formation.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ScriptAccess",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::*.amzn-s3-demo-bucket/scripts"
      ]
    },
    {
      "Sid": "LoggingAccess",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/logs/*"
      ]
    },
```

```
    {
      "Sid": "GlueCatalogAccess",
      "Effect": "Allow",
      "Action": [
        "glue:Get*",
        "glue:Create*",
        "glue:Update*"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "LakeFormationAccess",
      "Effect": "Allow",
      "Action": [
        "lakeformation:GetDataAccess"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Step 2.1 Configure Lake Formation permissions

- Spark jobs that read data from S3 require Lake Formation SELECT permission.
- Spark jobs that write/delete data in S3 require Lake Formation ALL (SUPER) permission.
- Spark jobs that interact with Glue Data catalog require DESCRIBE, ALTER, DROP permission as appropriate.

For more information, see Granting permissions on Data Catalog resources.

Step 3: Initialize a Spark session for Full Table Access using Lake Formation

Prerequisites

AWS Glue Data Catalog must be configured as a metastore to access Lake Formation tables.

Set the following settings to configure Glue catalog as a metastore:

```
--conf spark.sql.catalogImplementation=hive
--conf
 spark.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSGlueDataCata
```

To enable Data Catalog for EMR on EC2, see Use AWS Glue Data Catalog with Spark on Amazon EMR.

Accessing tables

To access tables registered with AWS Lake Formation, the following configurations need to be set during Spark initialization to configure Spark to use AWS Lake Formation credentials.

Hive

```
--conf
 spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.glue.accesscontrol.AWSLakeFormati
--conf spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true
--conf spark.hadoop.fs.s3.folderObject.autoAction.disabled=true
--conf spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true
--conf spark.sql.catalog.createDirectoryAfterTable.enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
```

Iceberg

```
--conf spark.sql.catalog.spark_catalog=org.apache.iceberg.spark.SparkSessionCatalog
--conf spark.sql.catalog.spark_catalog.warehouse=S3_DATA_LOCATION
--conf spark.sql.catalog.spark_catalog.client.region=REGION
--conf spark.sql.catalog.spark_catalog.type=glue
--conf spark.sql.catalog.spark_catalog.glue.account-id=ACCOUNT_ID
--conf spark.sql.catalog.spark_catalog.glue.lakeformation-enabled=true
--conf spark.sql.catalog.dropDirectoryBeforeTable.enabled=true
```

- `spark.hadoop.fs.s3.credentialsResolverClass=com.amazonaws.glue.accesscontrol.AW` Configure EMR Filesystem (EMRFS) to use AWS Lake Formation S3 credentials for Lake Formation registered tables. If the table is not registered, use the job's runtime role credentials.
- `spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true` and `spark.hadoop.fs.s3.folderObject.autoAction.disabled=true`: Configure EMRFS to use content type header application/x-directory instead of `$folder$` suffix when creating S3

folders. This is required when reading Lake Formation tables, as Lake Formation credentials do not allow reading table folders with $folder$ suffix.

- `spark.sql.catalog.skipLocationValidationOnCreateTable.enabled=true`: Configure Spark to skip validating the table location's emptiness before creation. This is necessary for Lake Formation registered tables, as Lake Formation credentials to verify the empty location are available only after Glue Data Catalog table creation. Without this configuration, the job's runtime role credentials will validate the empty table location.
- `spark.sql.catalog.createDirectoryAfterTable.enabled=true`: Configure Spark to create the Amazon S3 folder after table creation in the Hive metastore. This is required for Lake Formation registered tables, as Lake Formation credentials to create the S3 folder are available only after Glue Data Catalog table creation.
- `spark.sql.catalog.dropDirectoryBeforeTable.enabled=true`: Configure Spark to drop the S3 folder before table deletion in the Hive metastore. This is necessary for Lake Formation registered tables, as Lake Formation credentials to drop the S3 folder are not available after table deletion from the Glue Data Catalog.
- `spark.sql.catalog.<catalog>.glue.lakeformation-enabled=true`: Configure Iceberg catalog to use AWS Lake Formation S3 credentials for Lake Formation registered tables. If the table is not registered, use default environment credentials.

Configure full table access mode in SageMaker Unified Studio

To access Lake Formation registered tables from interactive Spark sessions in JupyterLab notebooks, you must use compatibility permission mode. Use the `%%configure` magic command to set up your Spark configuration. Choose the configuration based on your table type:

For Hive tables

```
%%configure -f
{
    "conf": {
        "spark.hadoop.fs.s3.credentialsResolverClass":
 "com.amazonaws.glue.accesscontrol.AWSLakeFormationCredentialResolver",
        "spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject": true,
        "spark.hadoop.fs.s3.folderObject.autoAction.disabled": true,
        "spark.sql.catalog.skipLocationValidationOnCreateTable.enabled": true,
        "spark.sql.catalog.createDirectoryAfterTable.enabled": true,
        "spark.sql.catalog.dropDirectoryBeforeTable.enabled": true
    }
```

```
}
```

For Iceberg tables

```
%%configure -f
{
    "conf": {
        "spark.sql.catalog.spark_catalog":
 "org.apache.iceberg.spark.SparkSessionCatalog",
        "spark.sql.catalog.spark_catalog.warehouse": "S3_DATA_LOCATION",
        "spark.sql.catalog.spark_catalog.client.region": "REGION",
  "conf spark.sql.catalog.spark_catalog.type": "glue",
        "spark.sql.catalog.spark_catalog.glue.account-id": "ACCOUNT_ID",
        "spark.sql.catalog.spark_catalog.glue.lakeformation-enabled": true,
        "spark.sql.catalog.dropDirectoryBeforeTable.enabled": true
    }
}
```

Replace the placeholders:

- `S3_DATA_LOCATION`: Your S3 bucket path
- `REGION`: AWS region (e.g., us-east-1)
- `ACCOUNT_ID`: Your AWS account ID

> **Note**
>
> You must set these configurations before executing any Spark operations in your notebook.

Supported Operations

These operations will use AWS Lake Formation credentials to access the table data.

- CREATE TABLE
- ALTER TABLE
- INSERT INTO
- INSERT OVERWRITE
- UPDATE

- MERGE INTO
- DELETE FROM
- ANALYZE TABLE
- REPAIR TABLE
- DROP TABLE
- Spark datasource queries
- Spark datasource writes

> **Note**
>
> Operations not listed above will continue to use IAM permissions to access table data.

Considerations

- If a Hive table is created using a job that doesn't have full table access enabled, and no records are inserted, subsequent reads or writes from a job with full table access will fail. This is because EMR Spark without full table access adds the `$folder$` suffix to the table folder name. To resolve this, you can either:
  - Insert at least one row into the table from a job that does not have FTA enabled.
  - Configure the job that does not have FTA enabled to not use `$folder$` suffix in folder name in S3. This can be achieved by setting Spark configuration `spark.hadoop.fs.s3.useDirectoryHeaderAsFolderObject=true`.
  - Create a S3 folder at the table location `s3://path/to/table/table_name` using the AWS S3 console or AWS S3 CLI.
- Full Table Access works exclusively with EMR Filesystem (EMRFS). S3A filesystem is not compatible.
- Full Table Access is supported for Hive and Iceberg tables. Support for Hudi and Delta tables has not yet been added.
- Jobs referencing tables with Lake Formation Fine-Grained Access Control (FGAC) rules or Glue Data Catalog Views will fail.
- Full table access does not support Spark Streaming.
- When writing Spark DataFrame to a Lake Formation table, only APPEND mode is supported: `df.write.mode("append").saveAsTable(table_name)`

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 with Amazon EMR

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.

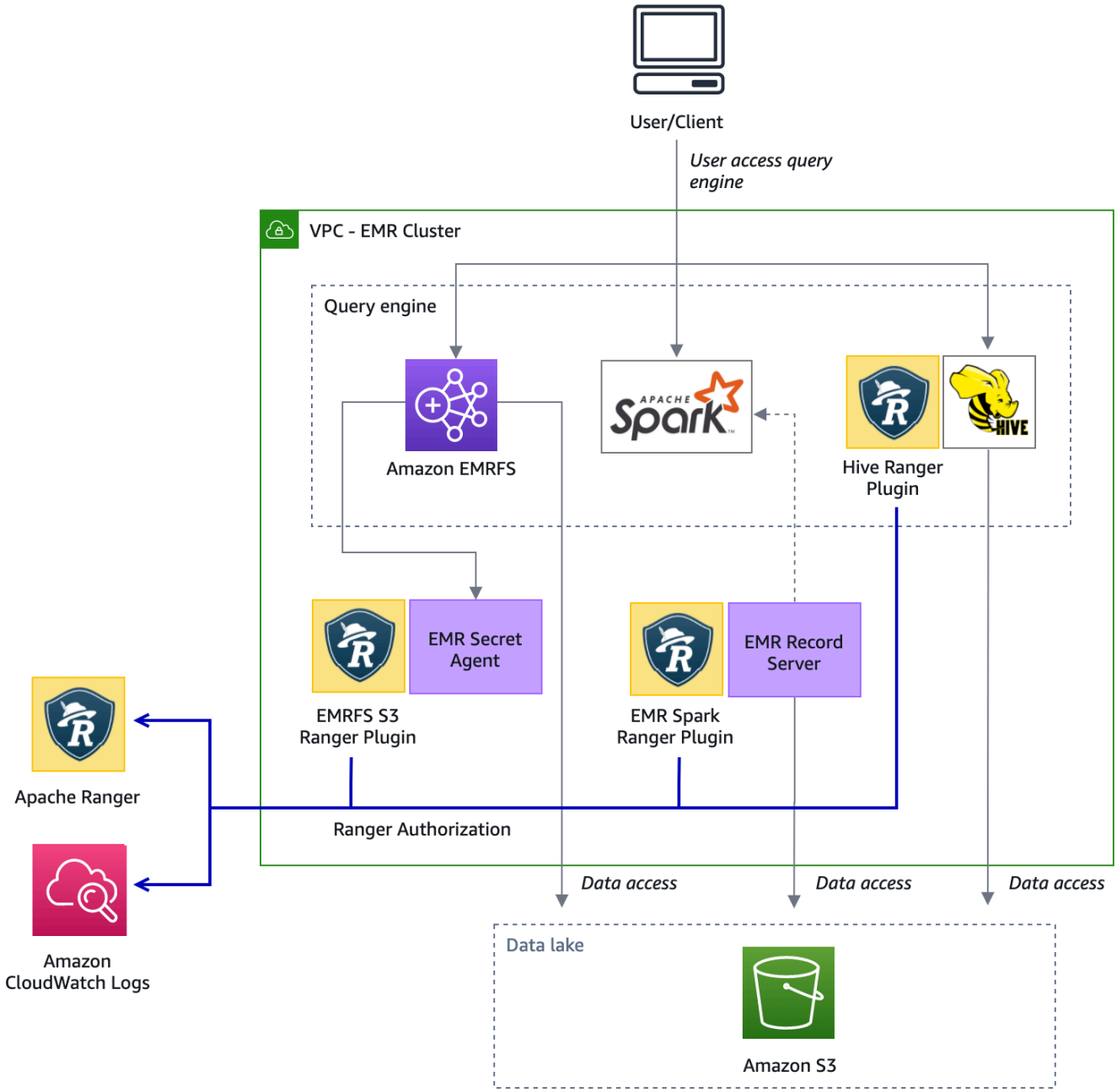
Apache Ranger uses two key components for authorization:

- **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
- Amazon EMR components for use with Apache Ranger

Architecture of Amazon EMR integration with Apache Ranger



Amazon EMR components for use with Apache Ranger

Amazon EMR enables fine-grained access control with Apache Ranger through the following components. See the architecture diagram 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.

Considerations for using Amazon EMR with Apache Ranger

Supported applications for Amazon EMR with Apache Ranger

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 of your cluster.
> AWS Glue Data Catalog and Open table formats such as Apache Hudi, Delta Lake, and Apache Iceberg aren't supported.

Supported Amazon EMR features with Apache Ranger

The following Amazon EMR features are supported when you use Amazon EMR with 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.

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.

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 a Ranger Admin server to integrate with Amazon EMR
- IAM roles for native integration with Apache Ranger

- Create the EMR security configuration
- Store TLS certificates in AWS Secrets Manager
- Start an EMR cluster with Apache Ranger
- Configure Zeppelin for Apache Ranger-enabled Amazon EMR clusters
- Known issues for Amazon EMR integration

Set up a Ranger Admin server to integrate with Amazon EMR

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 for Apache Ranger integration with Amazon EMR

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 for Ranger integration with Amazon EMR

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 for Ranger integration with Amazon EMR.

Installing EMRFS Service Definition

To install the S3 service definition for Amazon EMR, see EMRFS S3 plugin for Ranger integration with Amazon EMR.

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 for Ranger integration with Amazon EMR.

Network traffic rules for integrating with Amazon EMR

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 a Ranger Admin server to integrate with Amazon EMR.

EC2 instance profile for Amazon EMR

Amazon EMR uses an IAM service role 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) and Customize IAM roles with Amazon EMR.

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 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.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CloudwatchLogsPermissions",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:logs:*:123456789012:log-
group:CLOUDWATCH_LOG_GROUP_NAME_IN_SECURITY_CONFIGURATION:*"
      ]
    },
```

```
    {
      "Sid": "BucketPermissionsInS3Buckets",
      "Action": [
        "s3:CreateBucket",
        "s3:DeleteBucket",
        "s3:ListAllMyBuckets",
        "s3:ListBucket"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket1",
        "arn:aws:s3:::amzn-s3-demo-bucket2"
      ]
    },
    {
      "Sid": "ObjectPermissionsInS3Objects",
      "Action": [
        "s3:GetObject",
        "s3:DeleteObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket1/*",
        "arn:aws:s3:::amzn-s3-demo-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 for Amazon EMR integration

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 for Amazon EMR integration with Apache Ranger

See Apache Ranger troubleshooting 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.
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 Secrets Manager ARN for the Admin server and the address.
7. Select the applications to configure Ranger plugins. Enter the Secrets Manager ARN that contains 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.

> **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 CERTIFICATE
 WITHOUT VERSION>_",
         "RangerPluginConfigurations":[
            {
               "App":"Spark",
               "ClientSecretARN":"arn:aws:secretsmanager:_<REGION>_:_<ACCOUNT
 ID>_:secret:_<SECRET NAME THAT PROVIDES SPARK PLUGIN PRIVATE TLS CERTIFICATE
 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 CERTIFICATE 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 CERTIFICATE
 WITHOUT VERSION>_",
                "PolicyRepositoryName":"<EMRFS S3 SERVICE NAME eg amazon-emr-emrfs>"
            },
        {
                "App":"Trino",
                "ClientSecretARN":"arn:aws:secretsmanager:_<REGION>_:_<ACCOUNT
 ID>_:secret:_<SECRET NAME THAT PROVIDES TRINO PLUGIN PRIVATE TLS CERTIFICATE
 WITHOUT VERSION>_",
                "PolicyRepositoryName":"<TRINO SERVICE NAME eg amazon-emr-trino>"
            }
         ],
         "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 for authentication with Amazon EMR.
- (Optional) Enable encryption in transit or at rest. For more information, see Encryption options for Amazon EMR.

For more information, see Security in Amazon EMR.

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 a Ranger Admin server to integrate with Amazon EMR. 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-----
...Certificate Body...
-----END CERTIFICATE-----
```

An example of a private certificate is in the format:

```
-----BEGIN PRIVATE KEY-----
...Private Certificate Body...
-----END PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
...Trust Certificate Body...
-----END CERTIFICATE-----
```

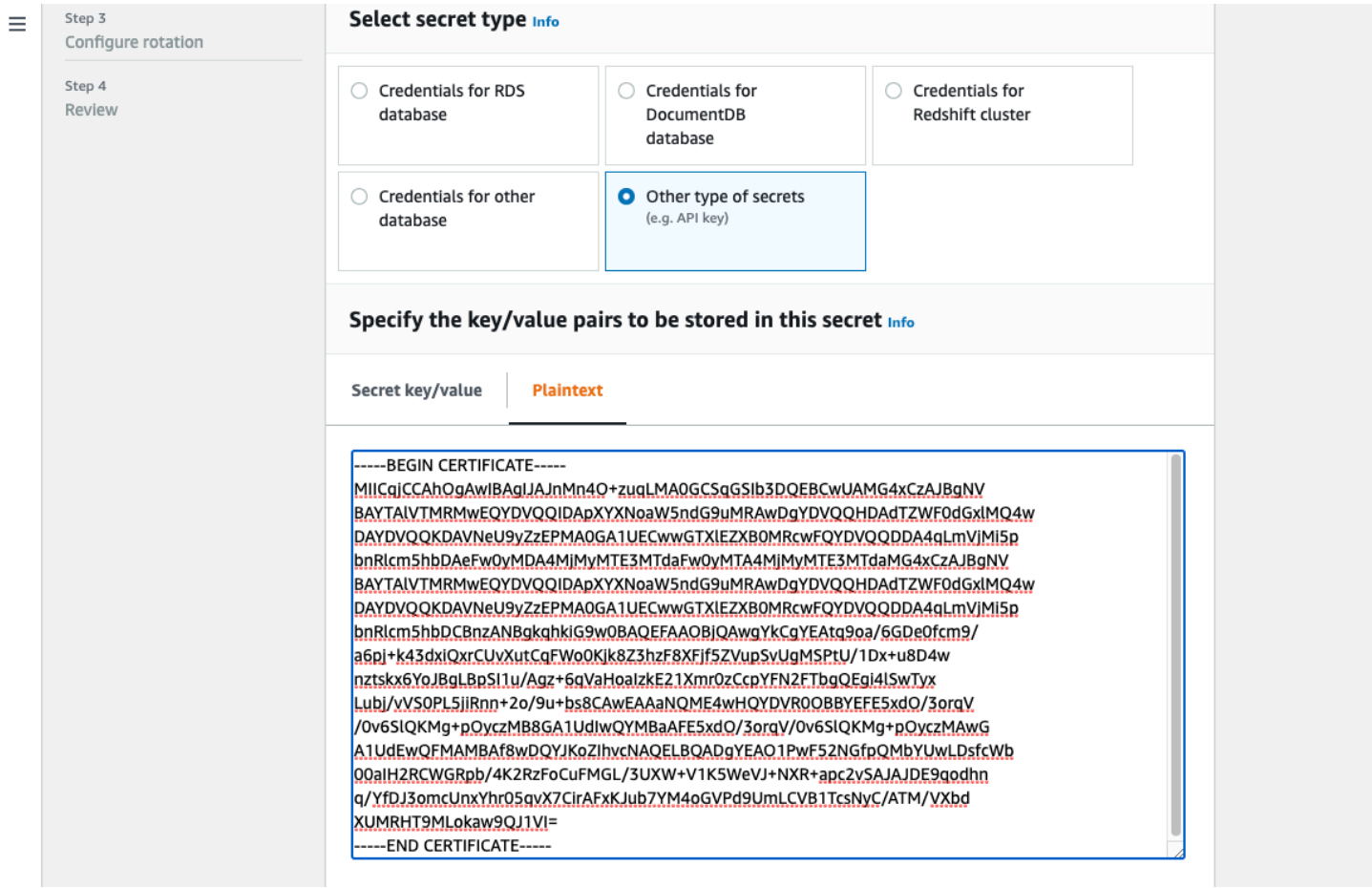
The private certificate should also contain a trust certificate as well.

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 with Apache Ranger

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 a Ranger Admin server to integrate with Amazon EMR
- Create the necessary IAM Roles. See IAM roles for native integration with Apache Ranger.

- Create a EMR security configuration for Apache Ranger installation. See more information, see Create the EMR security configuration.

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 for Amazon EMR integration

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 for Amazon EMR integration scenarios

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 for Ranger integration with Amazon EMR
- Apache Spark plugin for Ranger integration with Amazon EMR
- EMRFS S3 plugin for Ranger integration with Amazon EMR
- Trino plugin for Ranger integration with Amazon EMR

Apache Hive plugin for Ranger integration with Amazon EMR

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

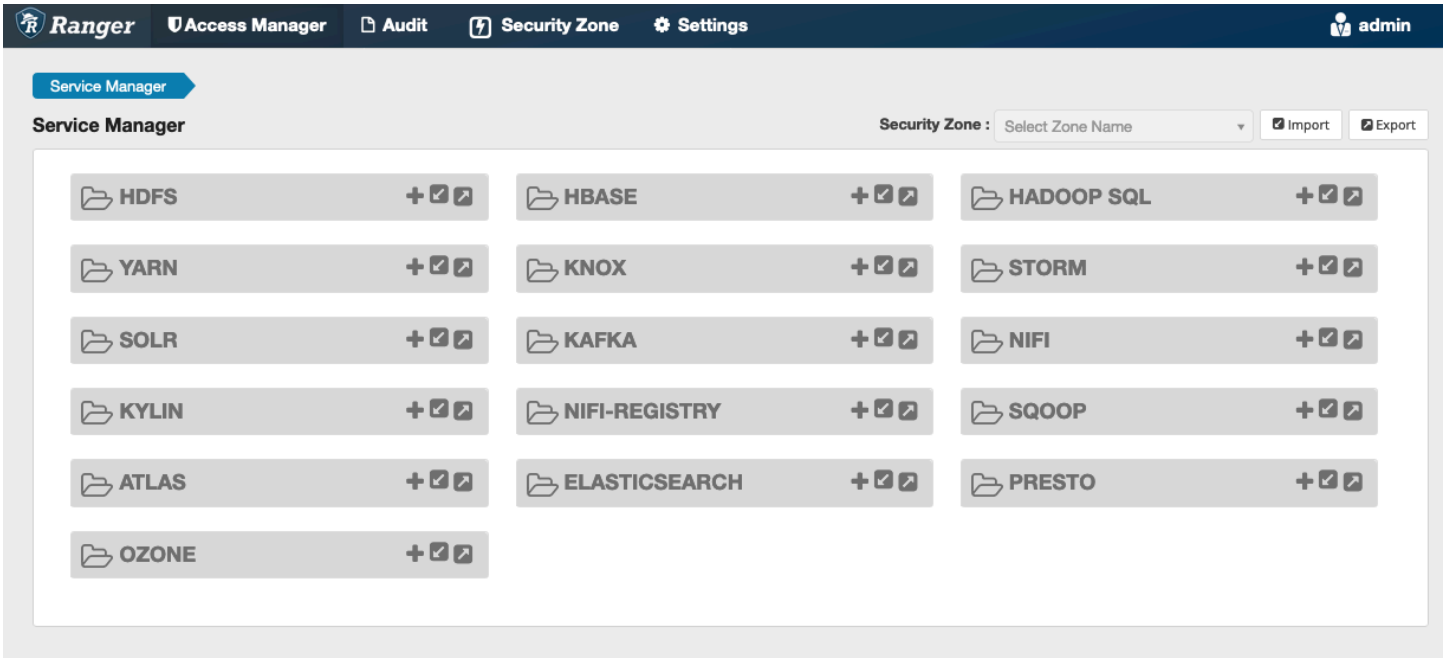
- Supported features
- Installation of service configuration
- Considerations
- Limitations

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.

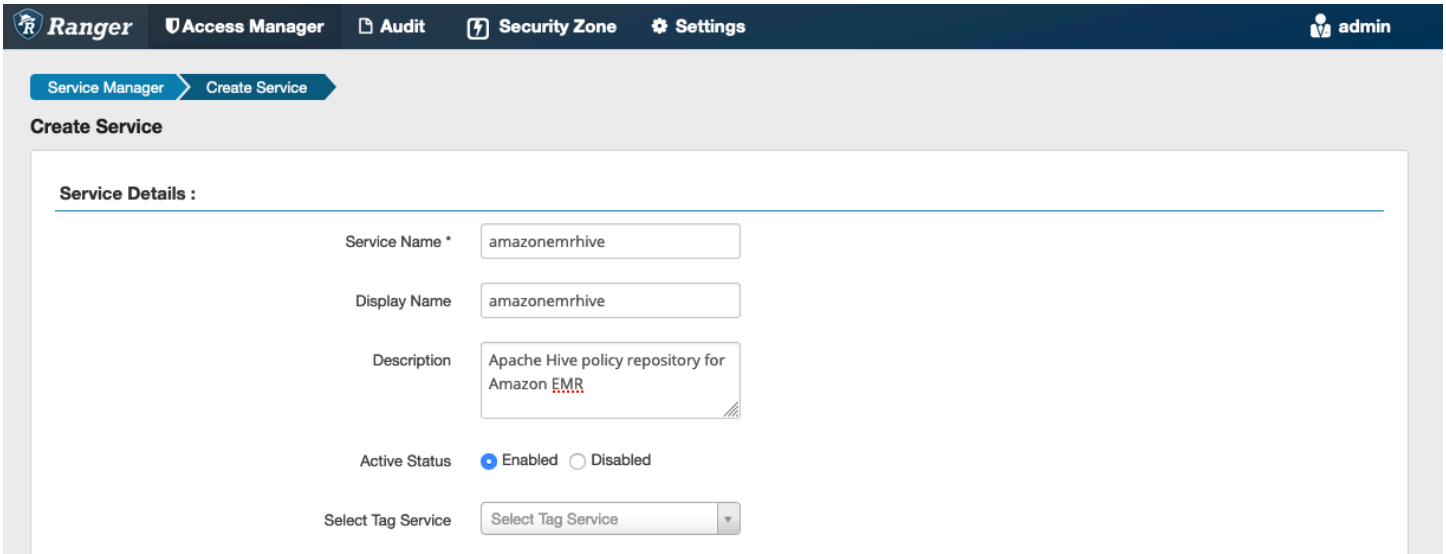
Installation of service configuration

The Apache Hive plugin is compatible with the existing Hive service definition within Apache Hive Hadoop SQL.



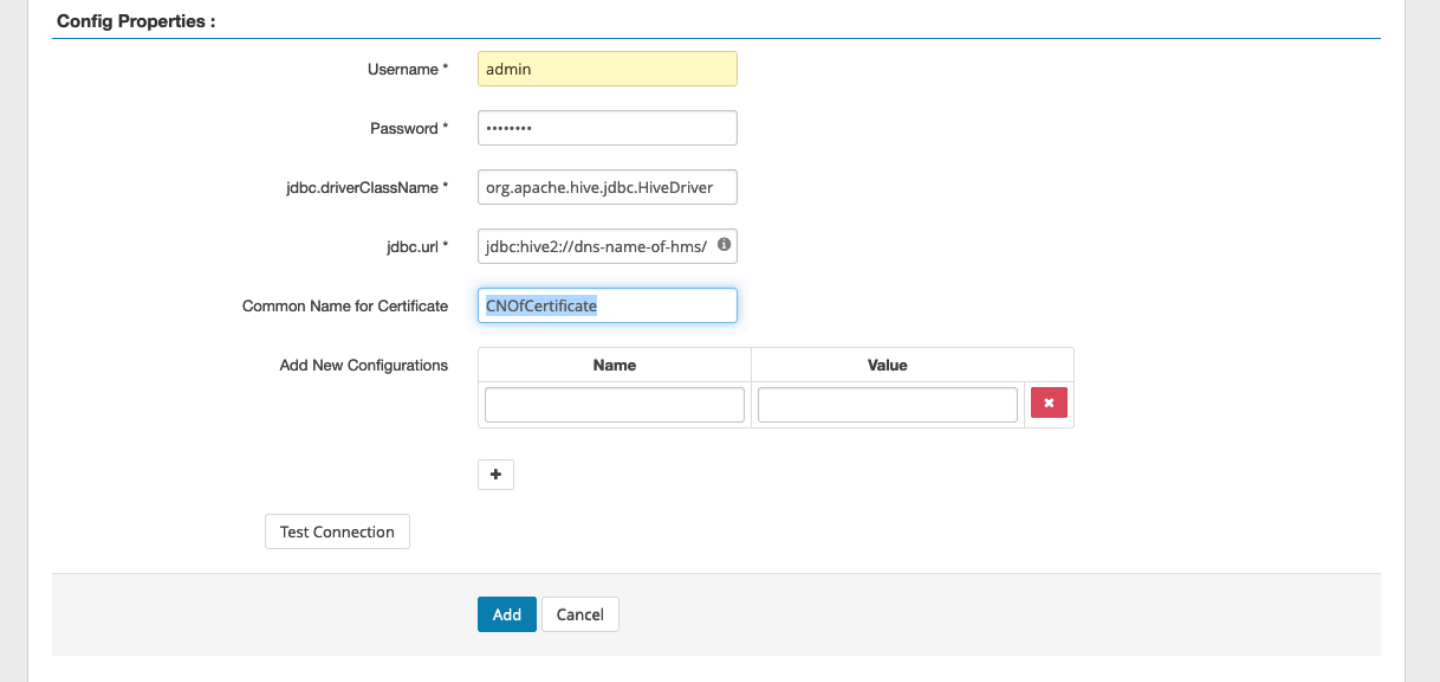
If you do not have an instance of the service under Hadoop SQL, like shown above, you can create one. Click on the **+** next to Hadoop SQL.

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`**.

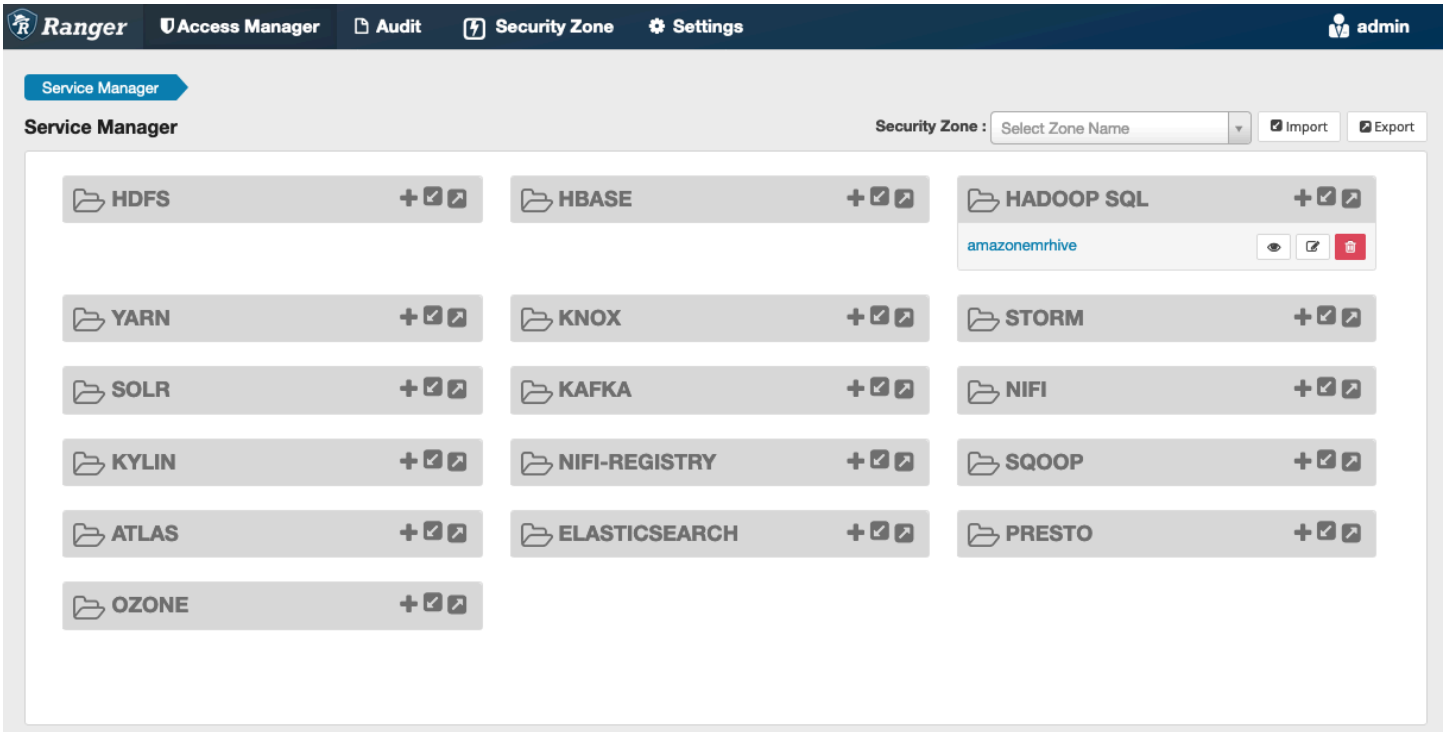


The Apache Hive Config Properties are used to establish a connection to your Apache Ranger Admin server with a HiveServer2 to implement auto complete when creating policies. The properties below are not required to be accurate if you do not have a persistent HiveServer2 process and can be filled with 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 for Ranger integration with Amazon EMR

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 for use with Apache Ranger page.

Topics

- Supported features
- Redeploy service definition to use INSERT, ALTER, or DDL statements
- Installation of service definition
- Creating SparkSQL policies
- Considerations
- Limitations

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

SQL statement/Ranger action	STATUS	Supported EMR release
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
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 a Ranger Admin server to integrate with Amazon EMR.

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
```

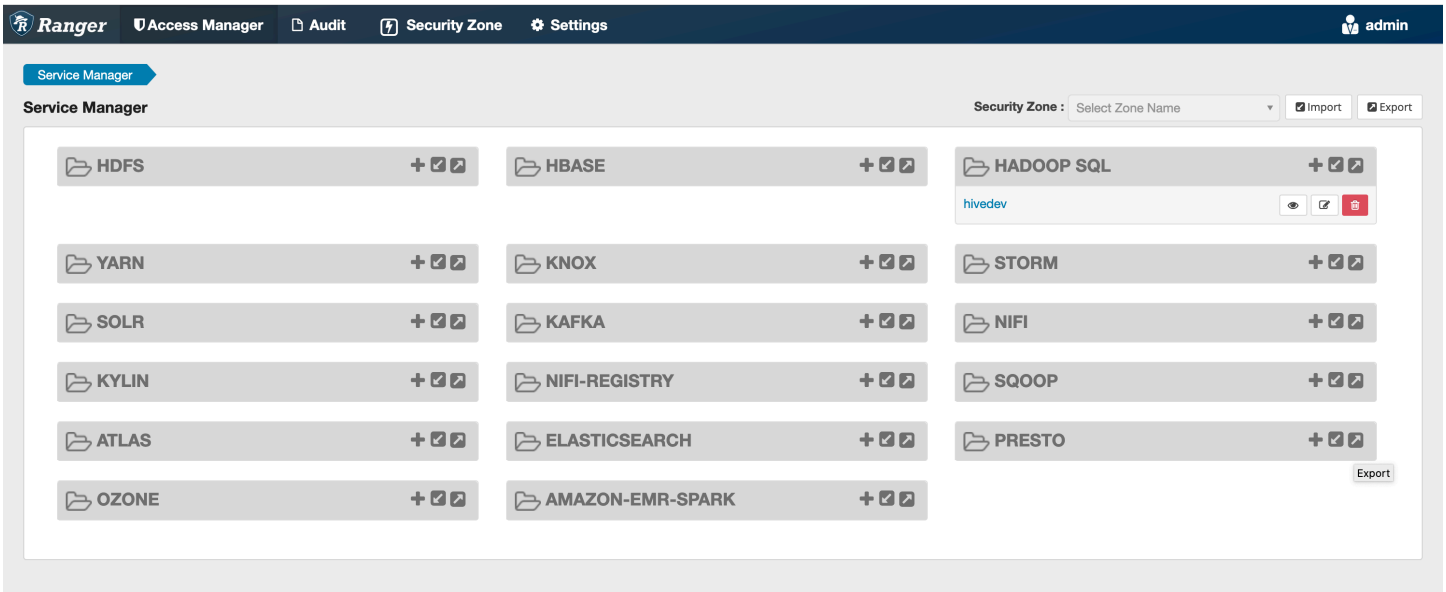
Step 3: Install the Apache Spark plugin for Amazon EMR

```
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).

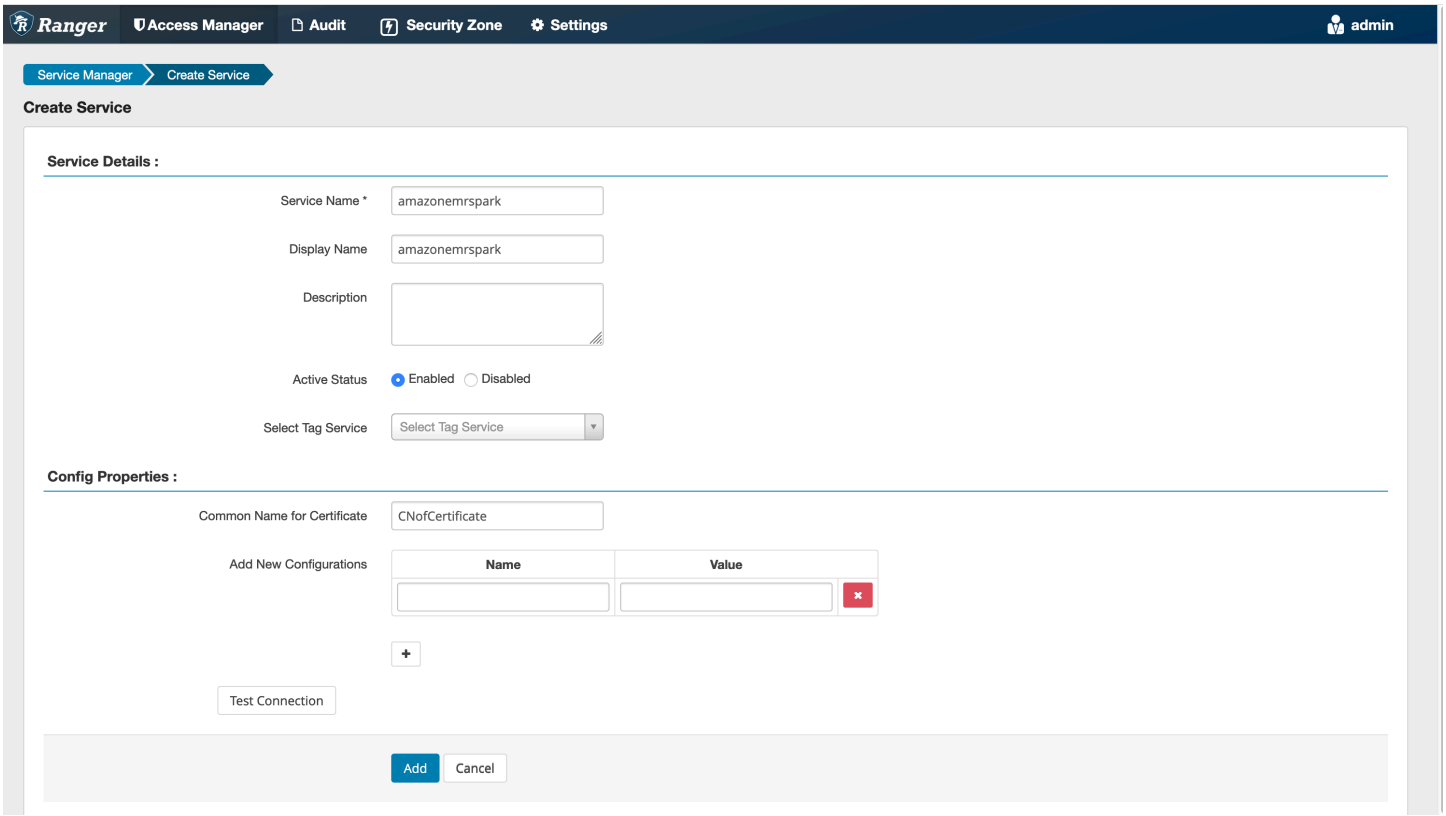


Step 5: Create an instance of the AMAZON-EMR-SPARK application

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 for Apache Ranger integration with Amazon EMR for more details.

Creating SparkSQL policies

When creating a new policy, the fields to fill in are:

Policy Name: The name of this policy.

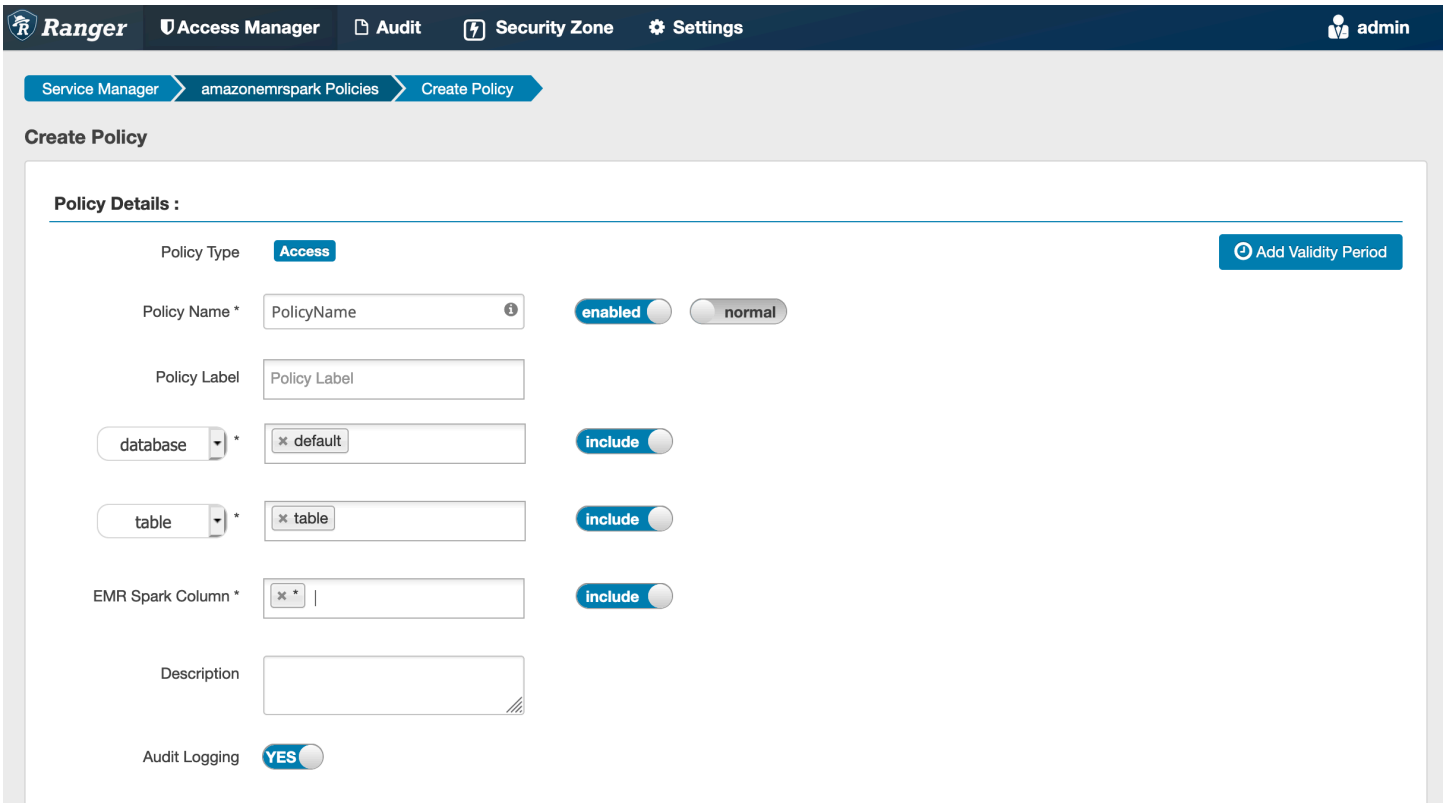
Policy Label: A label that you can put on this policy.

Database: The database that this policy applies to. The wildcard "*" represents all databases.

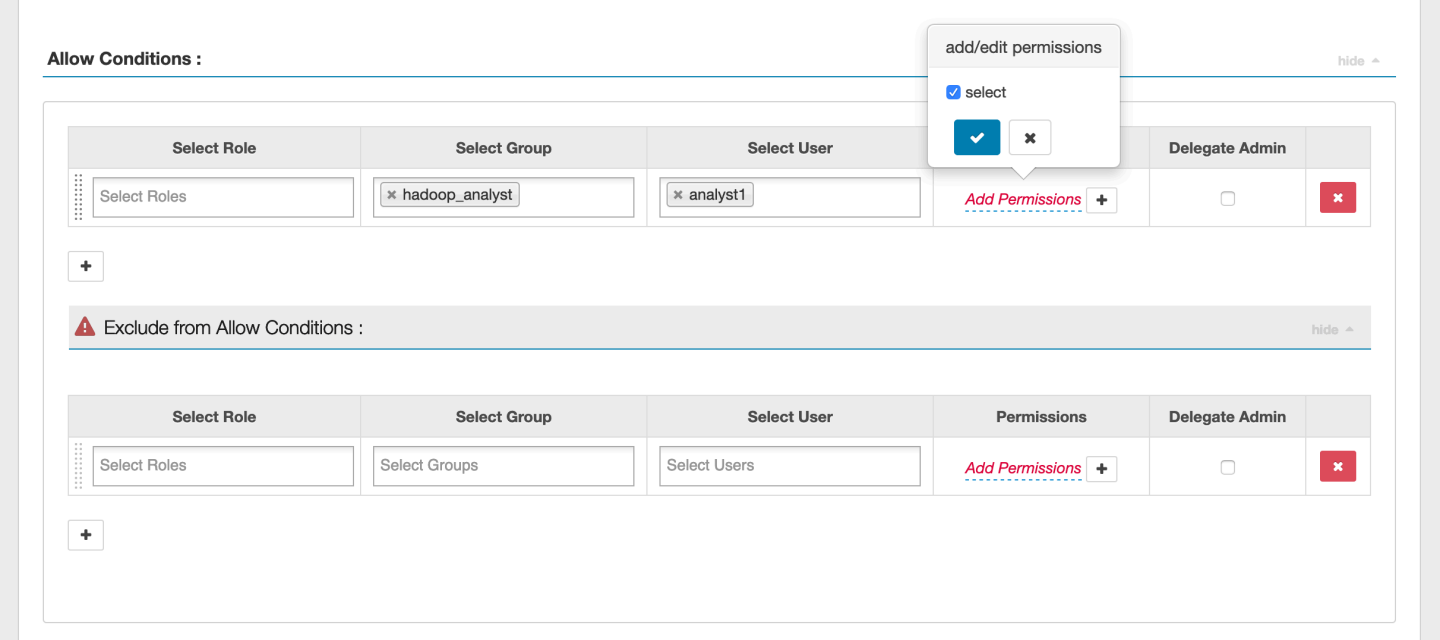
Table: The tables that this policy applies to. The wildcard "*" represents all tables.

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, Hudi and Iceberg tables aren't 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 for Ranger integration with Amazon EMR

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

- Installation of service configuration
- Creating EMRFS S3 policies
- EMRFS S3 policies usage notes
- Limitations

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

To install the EMRFS service definition, you must set up the Ranger Admin server. To set up the server, see Set up a Ranger Admin server to integrate with Amazon EMR.

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 EMRFS service definition.

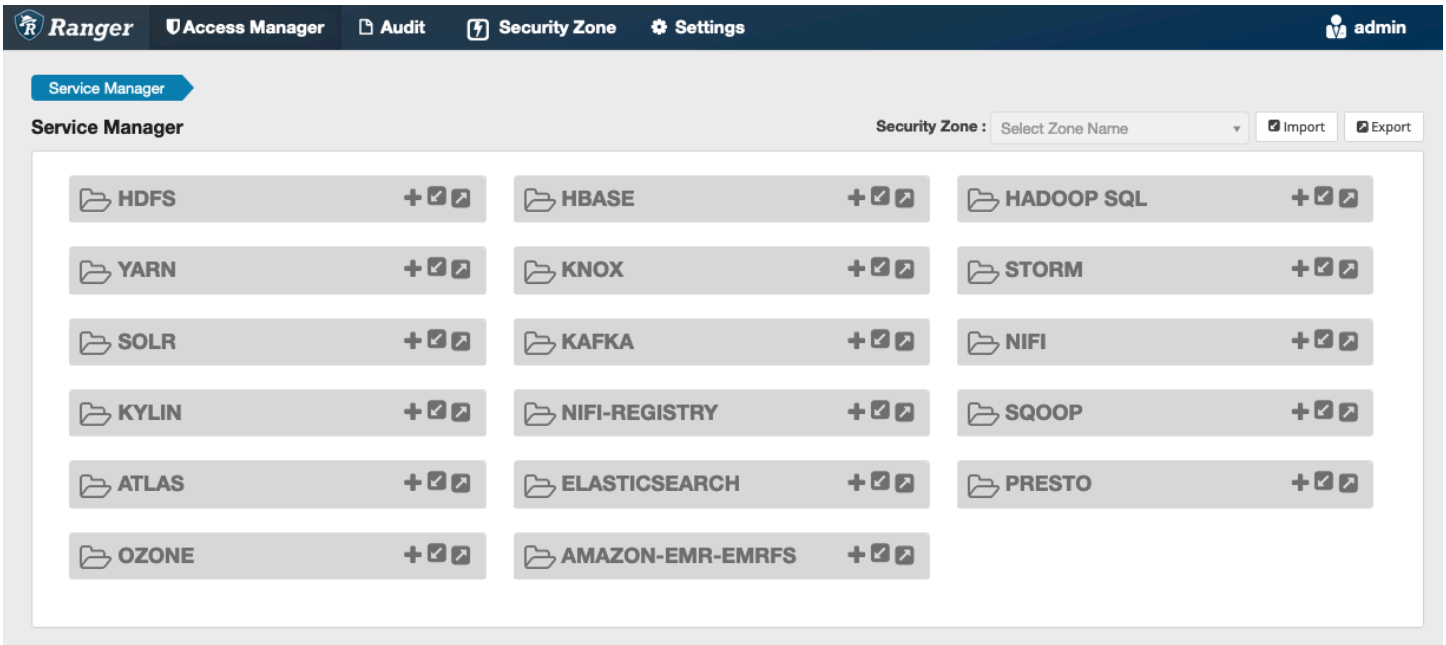
In a temporary directory, download the Amazon EMR service definition. This service definition is supported by Ranger 2.x versions.

```
wget https://s3.amazonaws.com/elasticmapreduce/ranger/service-definitions/version-2.0/
ranger-servicedef-amazon-emr-emrfs.json
```

Step 3: 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 4: Create an instance of the AMAZON-EMR-EMRFS application.

Create an instance of the service definition.

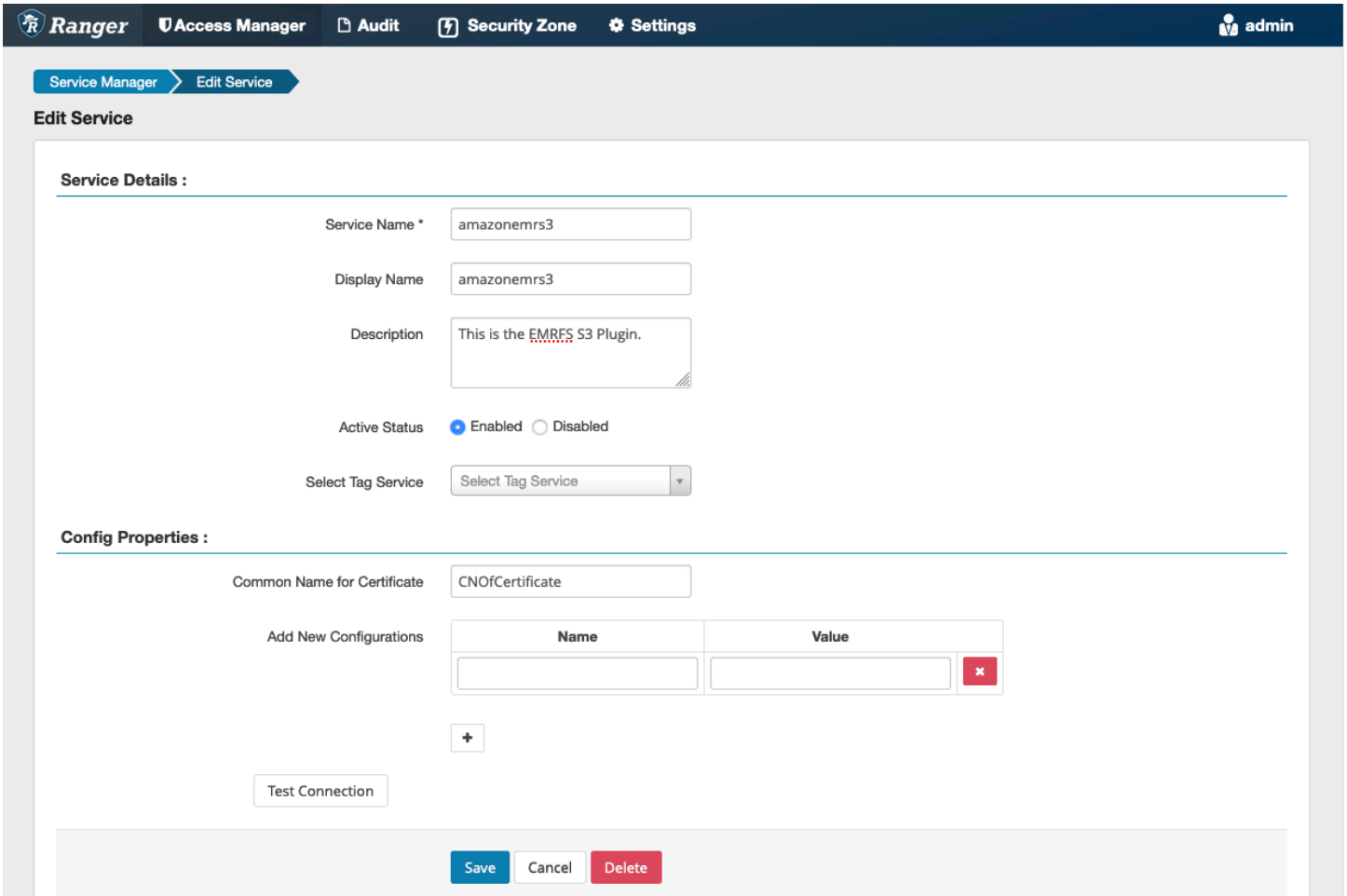
- Click on the **+** next to AMAZON-EMR-EMRFS.

Fill in the following fields:

Service Name (If displayed): The suggested value is **`amazonemrs3`**. Note this service name as it will be needed when creating an EMR security configuration.

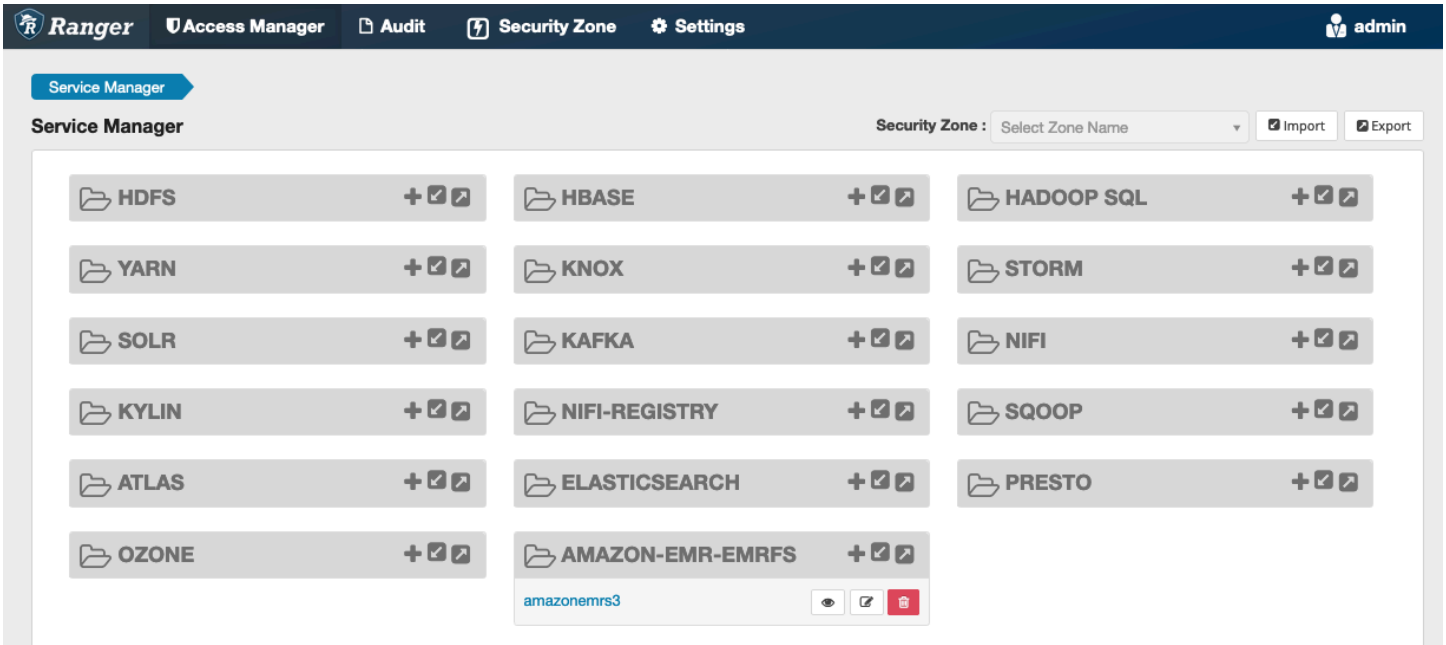
Display Name: The name displayed for this service. The suggested value is **`amazonemrs3`**.

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 for Apache Ranger integration with Amazon EMR for more details.

When the service is created, the Service Manager includes "AMAZON-EMR-EMRFS", as shown in the following image.



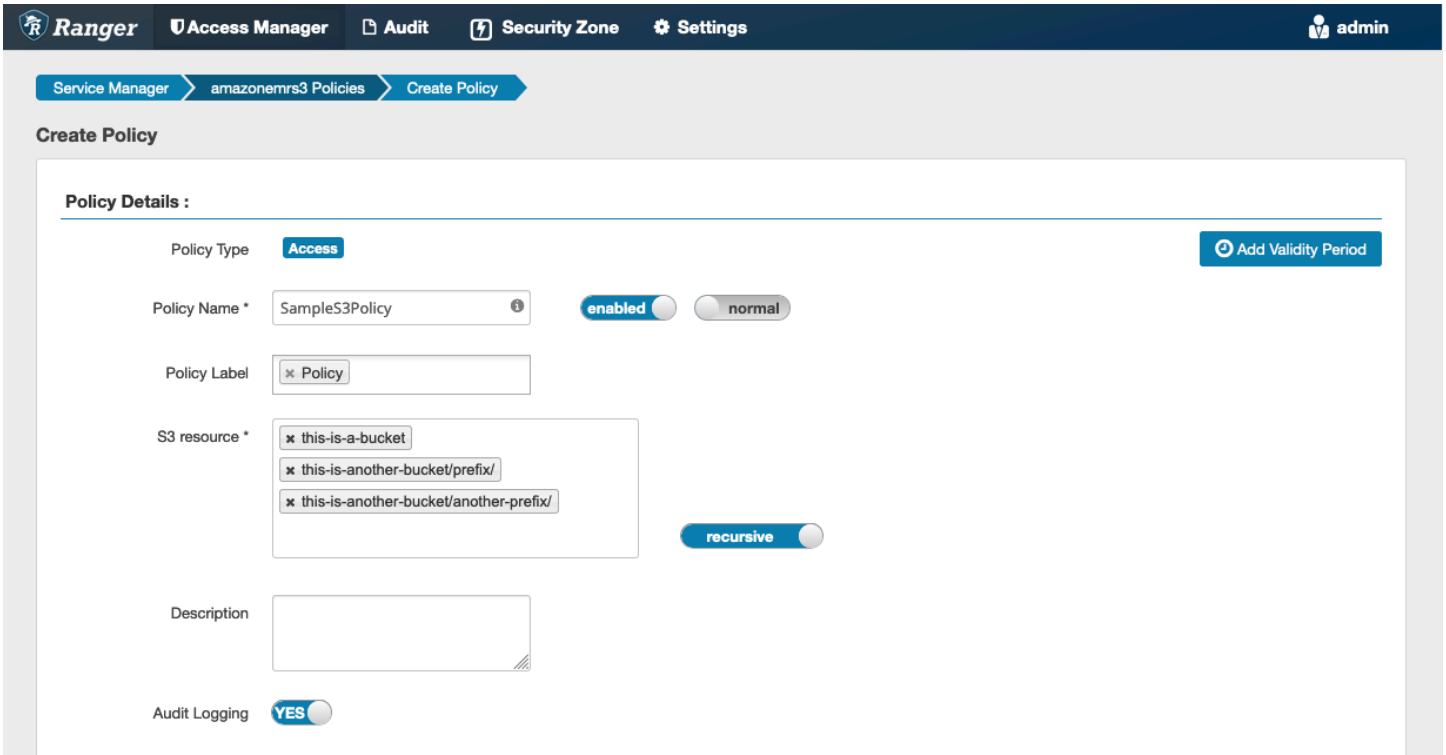
Creating EMRFS S3 policies

To create a new policy in the **Create policy** page of the Service Manager, fill in the following fields.

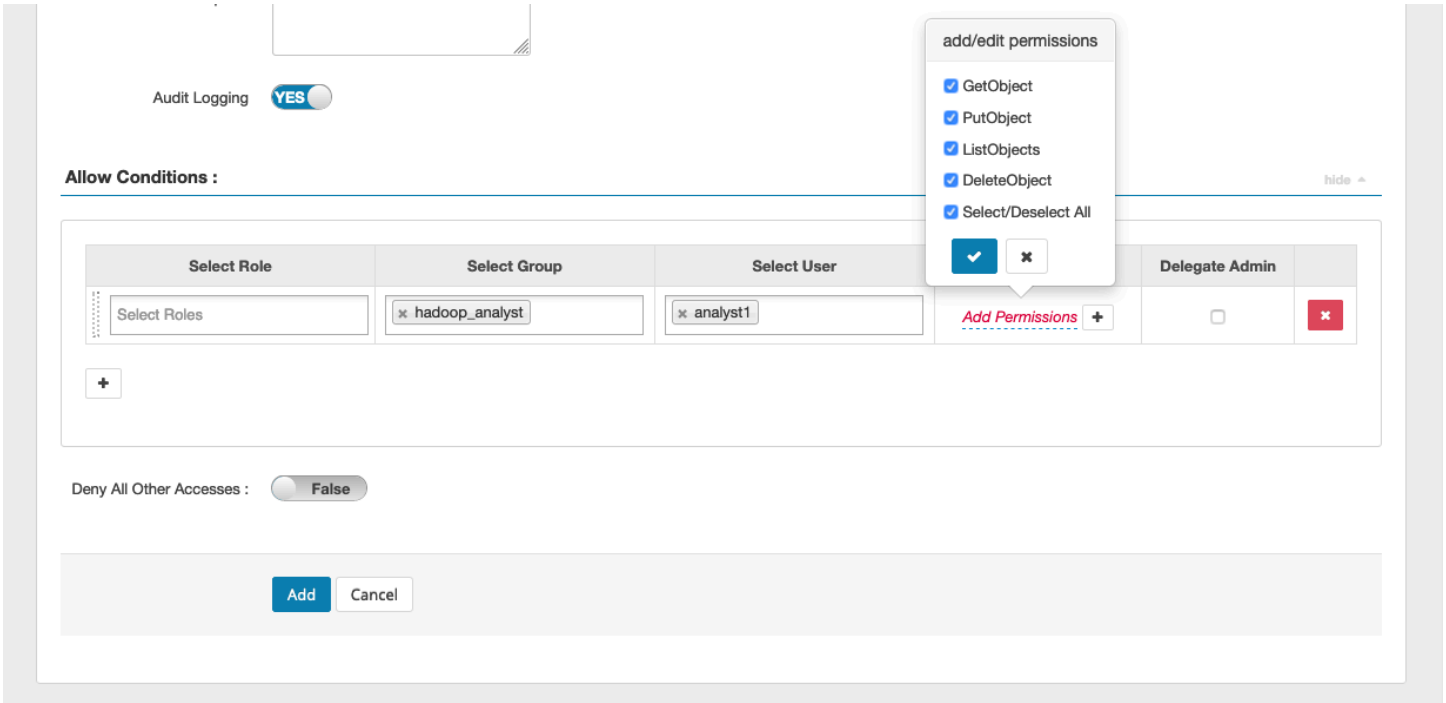
Policy Name: The name of this policy.

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 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.
- The Ranger Admin server doesn't support auto-complete.

Trino plugin for Ranger integration with Amazon EMR

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 wherever 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
- Installation of service configuration
- Creating Trino policies
- Considerations
- Limitations

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 a Ranger Admin server to integrate with Amazon EMR.

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'
```

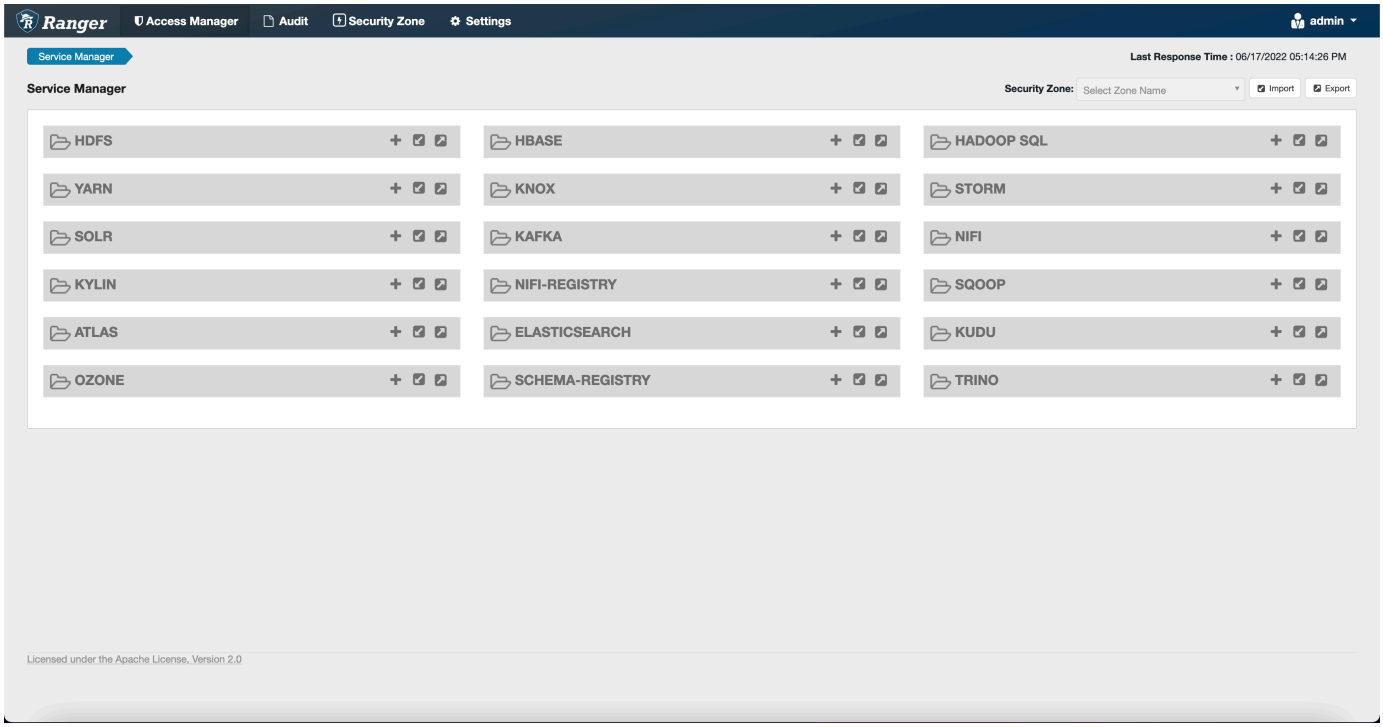
3. Download the service definition and Apache Ranger Admin server plugin. In a temporary directory, download the service definition. This service definition is supported by Ranger 2.x versions.

```
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'
```

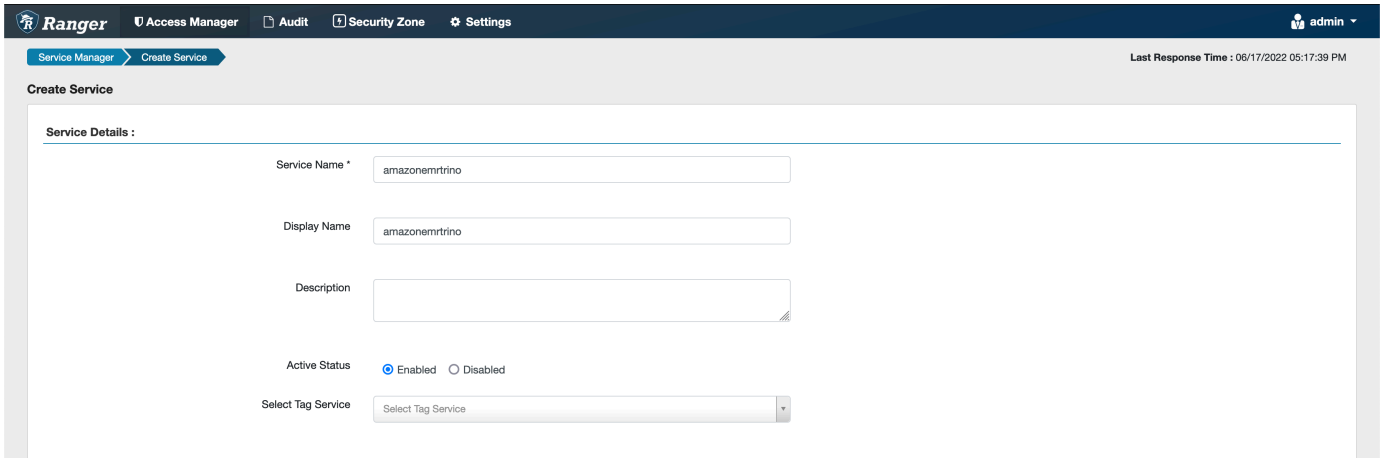
If this command runs successfully, you see a new service in your Ranger Admin UI called `TRINO`, as shown in the following image.



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.

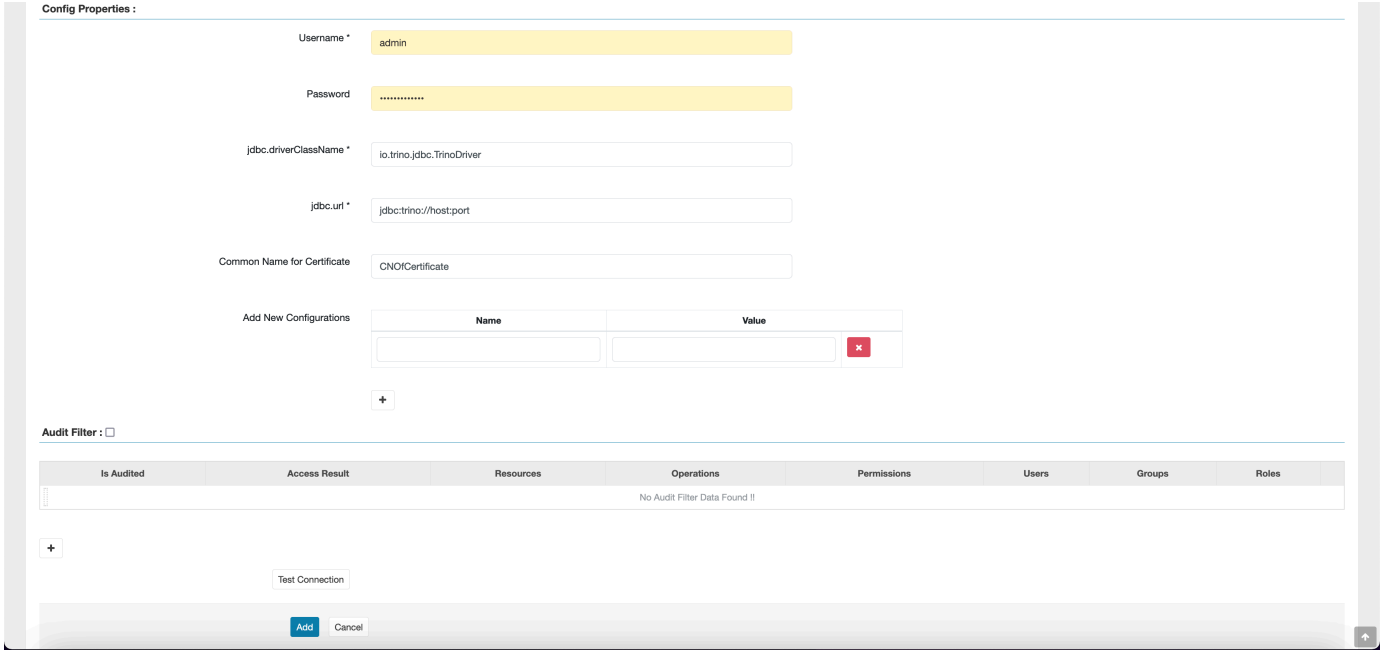
   Display Name: The name to be displayed for this instance. The suggested value is `amazonemrtrino`.



   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.

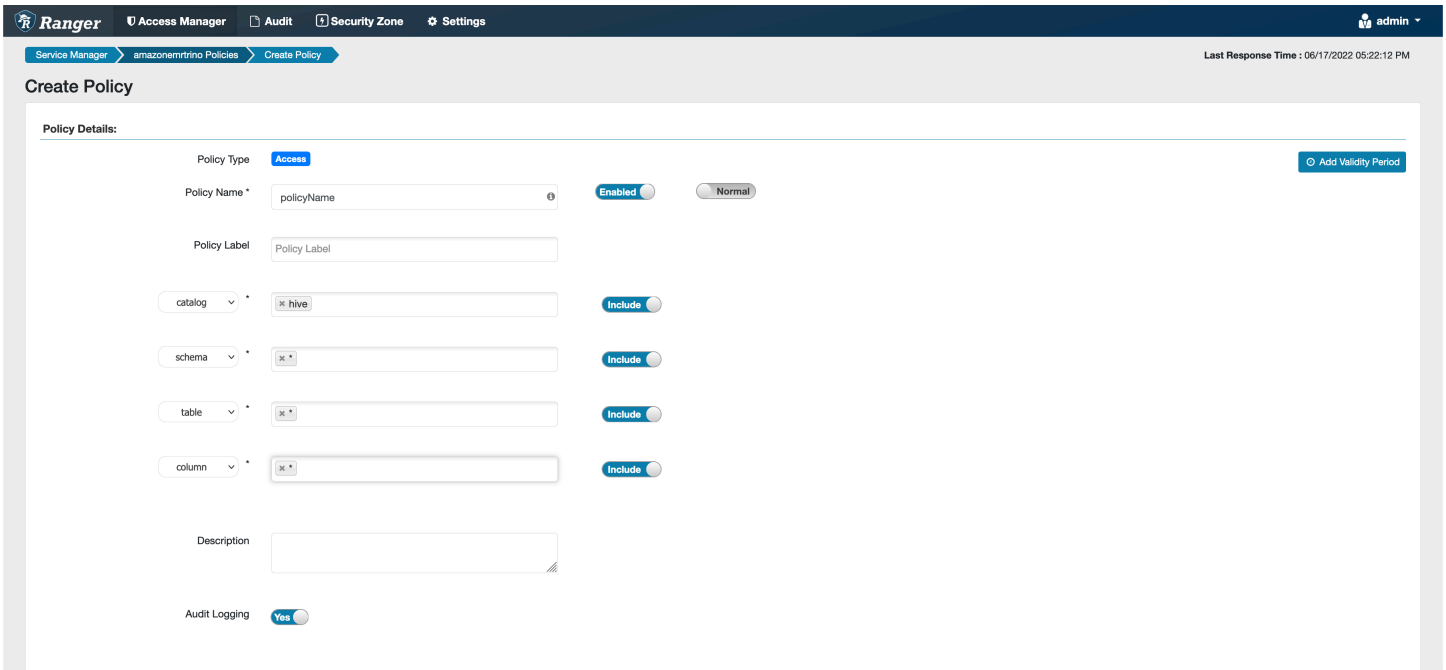
Schema: The schemas that this policy applies to. The wildcard "*" represents all schemas.

Table: The tables that this policy applies to. The wildcard "*" represents all tables.

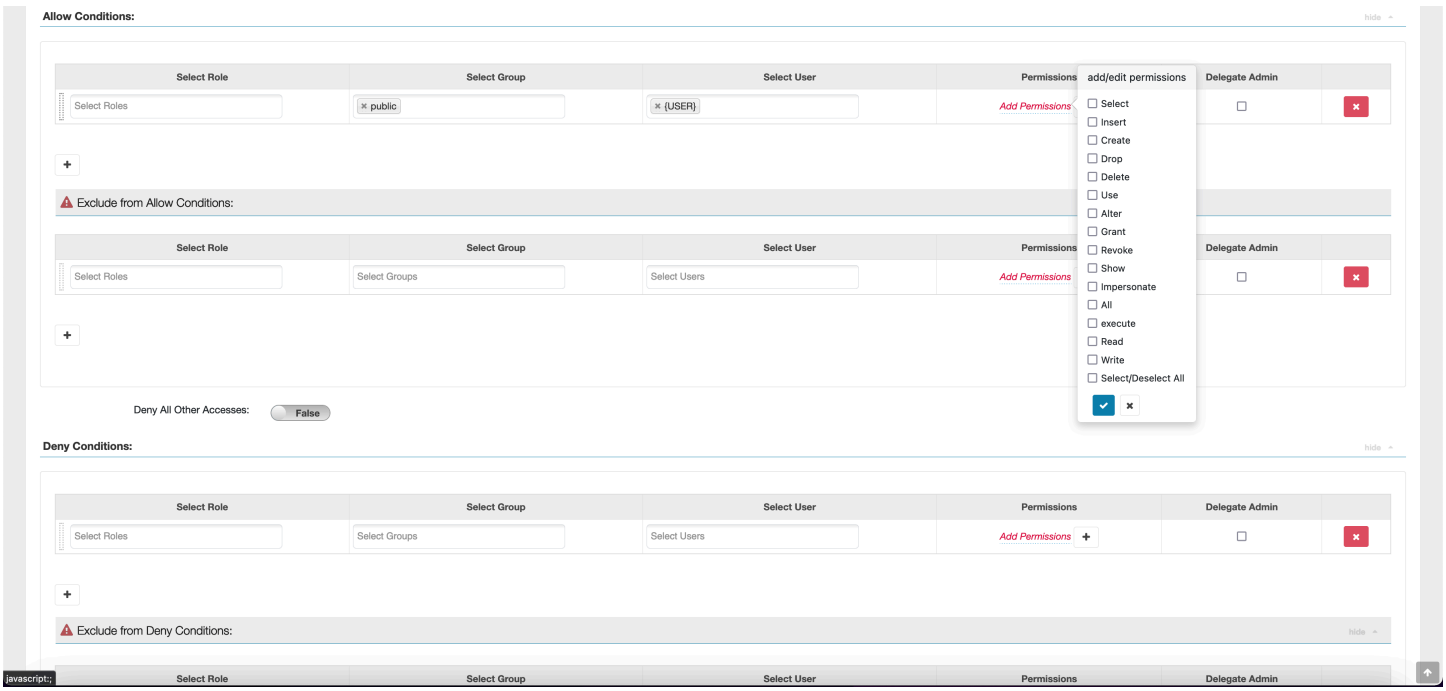
Column: The columns that this policy applies to. The wildcard "*" represents all columns.

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.

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/Ranger_plugins::Prepare_two_way_tls[configure 2-way TLS in Hive plugin]/Exec[create keystore and truststore	The private TLS certificate in Secret Manager for the Apache Ranger plugin certificate is not in the correct format or is not a private certificate. See TLS certificates for Apache

Error message	Cause
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	Ranger integration with Amazon EMR 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::XXXXXXXXXXXX: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-XXXXXXXXXXXXXX is	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.

Error message	Cause
not authorized to perform: sts:AssumeRole on resource: arn:aws:iam::XXXXXXXXXXXX :role/*RangerPluginDataAccessRole* (Service: AWSSecurityTokenService; Status Code: 403; Error Code: AccessDenied; Request ID: XXXX XXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX; Proxy: null)	
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.amazonaw s.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 for Ranger integration with Amazon EMR

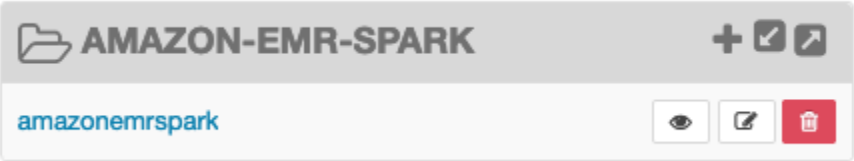
Check Apache Ranger plugin logs (Apache Hive, EMR RecordServer, EMR SecretAgent, etc., logs)

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 - [] Polic yRefresher(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 for your Amazon EMR cluster

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*.

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.

Topics

- Working with Amazon EMR-managed security groups
- Working with additional security groups for an Amazon EMR cluster
- Specifying Amazon EMR-managed and additional security groups
- Specifying EC2 security groups for EMR Notebooks
- Using Amazon EMR block public access

> **Note**
>
> Amazon EMR aims to use inclusive alternatives for potentially offensive or non-inclusive industry terms such as "master" and "slave". We've transitioned to new terminology to foster a more inclusive experience and to facilitate your understanding of the service components.
> We now describe "nodes" as **instances**, and we describe Amazon EMR instance types as **primary**, **core**, and **task** instances. During the transition, you might still find legacy references to the outdated terms, such as those that pertain to security groups for Amazon EMR.

Working with Amazon EMR-managed security groups

> **Note**
>
> Amazon EMR aims to use inclusive alternatives for potentially offensive or non-inclusive industry terms such as "master" and "slave". We've transitioned to new terminology to foster a more inclusive experience and to facilitate your understanding of the service components.
> We now describe "nodes" as **instances**, and we describe Amazon EMR instance types as **primary**, **core**, and **task** instances. During the transition, you might still find legacy references to the outdated terms, such as those that pertain to security groups for Amazon EMR.

Different managed security groups are associated with the primary 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 when you launch a cluster.

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 primary 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 Amazon EMR-managed security groups is also added to custom Amazon EMR-managed security groups that you specify.

The default managed security groups are as follows:

- **ElasticMapReduce-primary**

  For rules in this security group, see Amazon EMR-managed security group for the primary instance (public subnets).
- **ElasticMapReduce-core**

  For rules in this security group, see Amazon EMR-managed security group for core and task instances (public subnets).
- **ElasticMapReduce-Primary-Private**

  For rules in this security group, see Amazon EMR-managed security group for the primary instance (private subnets).
- **ElasticMapReduce-Core-Private**

  For rules in this security group, see Amazon EMR-managed security group for core and task instances (private subnets).
- **ElasticMapReduce-ServiceAccess**

For rules in this security group, see Amazon EMR-managed security group for service access (private subnets).

Amazon EMR-managed security group for the primary instance (public subnets)

The default managed security group for the primary instance in public subnets has the **Group Name** of **ElasticMapReduce-primary**. It has the following rules. If you specify a custom managed security group, Amazon EMR adds all the same rules to your custom security group.

Type	Protoc	Port range	Source	Details
Inbound rules				
All ICMP-IPv4	All	N/A	The Group ID of the managed security group for the primary 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-primary` for multiple clusters allows the core and task nodes of those clusters to communicate with each other over ICMP or any TCP or UDP port. Specify custom managed security groups to restrict cross-cluster access.
All TCP	TCP	All		
All UDP	UDP	All		
All ICMP-IPV4	All	N/A	The Group ID of the managed security group specified for core and task nodes.	These rules allow all inbound ICMP traffic and traffic over any TCP or UDP port from any core and task instances that are associated with the specified security group, even if the instances are in different clusters.
All TCP	TCP	All		
All UDP	UDP	All		
Custom	TCP	8443	Various Amazon IP address ranges	These rules allow the cluster manager to communicate with the primary node.

To grant trusted sources SSH access to the primary security group with the 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 a user and the Example Policy that allows managing EC2 security groups in the *IAM User Guide*.

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Choose **Clusters**. Choose the **ID** of the cluster you want to modify.
3. In the **Network and security** pane, expand the **EC2 security groups (firewall)** dropdown.
4. Under **Primary node**, choose your security group.
5. Choose **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, there was 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 primary 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 the other security group under **Core and task nodes** in the **Network and security** pane 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-core**. 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	Protoc	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-core` 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 primary instance.	These rules allow all inbound ICMP traffic and traffic over any TCP or UDP port from any primary 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 primary instance (private subnets)

The default managed security group for the primary instance in private subnets has the **Group Name** of **ElasticMapReduce-Primary-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	Protoc	Port range	Source	Details
Inbound rules				
All ICMP-IPv4	All	N/A	The Group ID of the managed security group for the primary 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-Primary-Private` for multiple clusters allows the core and task nodes of those clusters to communicate with each other over ICMP or any TCP or UDP port. Specify custom managed security groups to restrict cross-cluster access.
All TCP	TCP	All		
All UDP	UDP	All		
All ICMP-IPV4	All	N/A	The Group ID of the managed security group for core and task nodes.	These rules allow all inbound ICMP traffic and traffic over any TCP or UDP port from any core and task instances that are associated with the specified security group and reachable from within the private subnet, even if the instances are in different clusters.
All TCP	TCP	All		
All UDP	UDP	All		
HTTPS (8443)	TCP	8443	The Group ID of the managed security group for service access in a private subnet.	This rule allows the cluster manager to communicate with the primary node.

Type	Protoc	Port range	Source	Details
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 Amazon 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 Amazon 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-Core-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	Protoc	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-core` 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 primary instance.	These rules allow all inbound ICMP traffic and traffic over any TCP or UDP port from any primary 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 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 Amazon EMR 5.30.0 and later to connect to Amazon S3 over https.

Type	Protoc	Port range	Source	Details
				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. Amazon 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	Protoc	Port range	Destination	Details
All TCP	TCP	All	pl-*xxxxxxxx*	Managed Amazon S3 prefix list `com.amazonaws.`*`MyRegion`*`.s3`.
All Traffic	All	All	sg-*xxxxxxxxxxxxxxxx*	The ID of the `ElasticMapReduce-Core-Private` security group.
All Traffic	All	All	sg-*xxxxxxxxxxxxxxxx*	The ID of the `ElasticMapReduce-Primary-Private` security group.
Custom TCP	TCP	9443	sg-*xxxxxxxxxxxxxxxx*	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 primary node and with core and task nodes. The same rules are needed if you are using custom security groups.

Type	Protoc	Port range	Source	Details
Inbound rules Required for Amazon EMR clusters with Amazon EMR release 5.30.0 and later.				
Custom TCP	TCP	9443	The Group ID of the managed security group for primary instance.	This rule allows the communication between primary instance's security group to the service access security group.
Outbound rules Required for all Amazon EMR clusters				
Custom TCP	TCP	8443	The Group ID of the managed security group for primary instance.	These rules allow the cluster manager to communicate with the primary 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 primary node and with core and task nodes.

Working with additional security groups for an Amazon EMR cluster

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 primary 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 primary instance to each cluster in the subset.

You can apply up to 15 additional security groups for the primary instance, 15 for core and task instances, and 15 for service access (in private subnets). If necessary, you can specify the same additional security group for primary 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 Amazon 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 primary instances, core and task instances, and service access (private subnets only).

Console

To specify security groups with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose **Create cluster**.
3. Under **Networking**, select the arrow next to **EC2 security groups (firewall)** to expand this section. Under **Primary node** and **Core and task nodes**, the default Amazon EMR managed security groups are selected by default. If you use a private subnet, you also have the option to select a security group for **Service access**.
4. To change your Amazon EMR managed security group, use the **Choose security groups** dropdown menu to select a different option from the **Amazon EMR-managed security group** list of options. You have one Amazon EMR managed security group for both **Primary node** and **Core and task nodes**.
5. To add custom security groups, use the same **Choose security groups** dropdown menu to select up to four custom security groups from the **Custom security group** list of options.

You can have up to four custom security groups for both **Primary node** and **Core and task nodes**.

6. Choose any other options that apply to your cluster.
7. To launch your cluster, choose **Create cluster**.

Specifying security groups with 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
`EmrManagedPrimarySecurityGroup`	Use this parameter to specify a custom managed security group for the primary instance. If this parameter is specified, `EmrManagedCoreSecurityGroup` must also be specified. For clusters in private subnets, `ServiceAccessSecurityGroup` must also be specified.
`EmrManagedCoreSecurityGroup`	Use this parameter to specify a custom managed security group for core and task instances. If this parameter is specified, `EmrManagedPrimarySecurityGroup` 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.

Parameter	Description
	If this parameter is specified, `EmrManagedPrimarySecurityGroup` must also be specified.
`AdditionalPrimarySecurityGroups`	Use this parameter to specify up to four additional security groups for the primary instance.
`AdditionalCoreSecurityGroups`	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 primary 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-7.10.0 --applications Name=Hue Name=Hive \
Name=Pig --use-default-roles --ec2-attributes \
SubnetIds=subnet-xxxxxxxxxxxx,KeyName=myKey,\
ServiceAccessSecurityGroup=sg-xxxxxxxxxxxx,\
EmrManagedPrimarySecurityGroup=sg-xxxxxxxxxxxx,\
EmrManagedCoreSecurityGroup=sg-xxxxxxxxxxx,\
AdditionalPrimarySecurityGroups=['sg-xxxxxxxxxxx',\
'sg-xxxxxxxxxxx','sg-xxxxxxxxxx'],\
AdditionalCoreSecurityGroups=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 you use the notebook editor. 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 through TCP Port 18888. When you create custom security groups with rules that you tailor 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 cluster uses your custom security in addition to the default 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.

Default EC2 security group for the primary instance

The default EC2 security group for the primary instance is associated with the primary instance in addition to the cluster's security groups for the primary 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 so that the notebook can 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 (BPA)* prevents you from launching a cluster in a public subnet if the cluster has a security configuration that allows inbound traffic from public IP addresses on a port.

> **Important**
>
> *Block public access* is enabled by default. To increase account protection, we recommend that you keep it enabled.

Understanding block public access

You can use the *block public access* account-level configuration to centrally manage public network access to Amazon EMR clusters.

When a user from your AWS account launches a cluster, Amazon EMR checks the port rules in the security group for the cluster and compares them with your inbound traffic rules. If the security group has an inbound rule that opens ports to the public IP addresses IPv4 0.0.0.0/0 or IPv6 ::/0, and those ports aren't specified as exceptions for your account, Amazon EMR doesn't let the user create the cluster.

If a user modifies the security group rules for a running cluster in a public subnet to have a public access rule that violates the BPA configuration for your account, Amazon EMR revokes the new rule if it has permission to do so. If Amazon EMR doesn't have permission to revoke the rule, it creates an event in the AWS Health dashboard that describes the violation. To grant the revoke rule permission to Amazon EMR, see Configure Amazon EMR to revoke security group rules.

Block public access is enabled by default for all clusters in every AWS Region for your AWS account. BPA applies to the entire lifecycle of a cluster, but doesn't apply to clusters that you create in private subnets. You can configure exceptions to the BPA rule; port 22 is an exception by default. For more information on setting exceptions, see Configure block public access.

Configure block public access

You can update security groups and the block public access configuration in your accounts at any time.

You can turn block public access (BPA) settings on and off with the AWS Management Console, the AWS Command Line Interface (AWS CLI), and the Amazon EMR API. Settings apply across your account on a Region-by-Region basis. To maintain cluster security, we recommend that you use BPA.

Console

To configure block public access with the console

1. Sign in to the AWS Management Console, then open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. On the top navigation bar, select the **Region** that you want to configure if it's not already selected.
3. Under **EMR on EC2** in the left navigation pane, 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 **Edit**, choose **Turn on** or **Turn off** as appropriate, and then choose **Save**.
Edit ports in the list of exceptions	1. Choose **Edit** and find the **Port range exceptions** section. 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 **Remove** next to the entry in the list of port ranges. 4. Choose **Save**.

AWS CLI

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. `aws emr put-block-public-access-configuration --block-public-access-configuration BlockPublicSecurityGroupRules=false`

To...	Do this...
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 } ] }'`

Configure Amazon EMR to revoke security group rules

Amazon EMR needs permission to revoke security group rules and comply with your block public access configuration. You can use one of the following approaches to give Amazon EMR the permission that it needs:

- **(Recommended)** Attach the `AmazonEMRServicePolicy_v2` managed policy to the service role. For more information, see Service role for Amazon EMR (EMR role).
- Create a new inline policy that allows the `ec2:RevokeSecurityGroupIngress` action on security groups. For more information about how to modify a role permissions policy, see **Modifying a role permissions policy** with the IAM Console, AWS API, and AWS CLI in the *IAM User Guide*.

Resolve block public access violations

If a block public access violation occurs, you can mitigate it with one of the following actions:

- If you want to access a web interface on your cluster, use one of the options described in View web interfaces hosted on Amazon EMR clusters to access the interface through SSH (port 22).
- To allow traffic to the cluster from specific IP addresses rather than from the public IP address, add a security group rule. For more information, see Add rules to a security group in the *Amazon EC2 Getting Started Guide*.
- **(Not recommended)** You can configure Amazon EMR BPA exceptions to include the desired port or range of ports. When you specify a BPA exception, you introduce risk with an unprotected port. If you plan to specify an exception, you should remove the exception as soon as it's no longer needed. For more information, see Configure block public access.

Identify clusters associated with security group rules

You might need to identify all of the clusters that are associated with a given security group rule, or to find the security group rule for a given cluster.

- If you know the security group, then you can identify its associated clusters if you find the network interfaces for the security group. For more information, see How can I find the resources associated with an Amazon EC2 security group? on AWS re:Post. The Amazon EC2 instances that are attached to these network interfaces will be tagged with the ID of the cluster that they belong to.
- If you want to find the security groups for a known cluster, follow the steps in View Amazon EMR cluster status and details. You can find the security groups for the cluster in the **Network and security** panel in the console, or in the `Ec2InstanceAttributes` field from the AWS CLI.

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 AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To

design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access Amazon EMR through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). 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

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 a 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 Amazon EMR clusters

After you've launched your cluster, you can connect to it and manage it. Amazon EMR provides a collection of tools you can use to do this. This section provides guidance for connecting to your cluster, giving it work to do, monitoring work as it's done, and troubleshooting issues.

Topics

- Connect to an Amazon EMR cluster
- Submit work to an Amazon EMR cluster
- View and monitor an Amazon EMR cluster as it performs work
- Use Amazon EMR cluster scaling to adjust for changing workloads
- Terminate an Amazon EMR cluster in the starting, running, or waiting states
- Clone an Amazon EMR cluster using the console
- Automate recurring Amazon EMR clusters with AWS Data Pipeline

Connect to an Amazon EMR 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 primary node while the cluster is running. For example, you may want to connect to the primary 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 primary node, and so on. The following sections describe techniques that you can use to connect to the primary node.

In an EMR cluster, the primary node is an Amazon EC2 instance that coordinates the EC2 instances that are running as task and core nodes. The primary node exposes a public DNS name that you can use to connect to it. By default, Amazon EMR creates security group rules for the primary node, and for core and task nodes, that determine how you access the nodes.

> **Note**
>
> You can connect to the primary node only while the cluster is running. When the cluster terminates, the EC2 instance acting as the primary node is terminated and is no longer available. To connect to the primary 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 for authentication with Amazon EMR. 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*.

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 to Amazon EMR: Authorize inbound traffic
- Connect to the Amazon EMR cluster primary node using SSH

Before you connect to Amazon EMR: 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 for your Amazon EMR cluster.

Console

To grant trusted sources SSH access to the primary security group with the 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 a user and the Example Policy that allows managing EC2 security groups in the *IAM User Guide*.

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose the cluster that you want to update. This opens up the cluster details page. The **Properties** tab on this page will be pre-selected.
3. Under **Networking** in the **Properties** tab, select the arrow next to **EC2 security groups (firewall)** to expand this section. Under **Primary node**, select the security group link. This opens the EC2 console.
4. Choose the **Inbound rules** tab and then choose **Edit inbound rules**.
5. 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 primary node. We strongly recommend that you remove this inbound rule and restrict traffic to trusted sources.

6. Scroll to the bottom of the list of rules and choose **Add Rule**.
7. For **Type**, select **SSH**. This selection automatically enters **TCP** for **Protocol** and **22** for **Port Range**.
8. 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.
9. Choose **Save**.
10. Optionally return to Step 3, choose **Core and task nodes**, and repeat Steps 4 - 8. This grants core and task nodes SSH client access.

Connect to the Amazon EMR cluster primary 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 primary node of the cluster.

Using SSH to connect to the primary node gives you the ability to monitor and interact with the cluster. You can issue Linux commands on the primary 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 primary node. For more information, see View web interfaces hosted on Amazon EMR clusters.

To connect to the primary node using SSH, you need the public DNS name of the primary node. In addition, the security group associated with the primary 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 for your Amazon EMR cluster and Adding rules to a security group in the *Amazon EC2 User Guide*.

Retrieve the public DNS name of the primary node

You can retrieve the primary public DNS name using the Amazon EMR console and the AWS CLI.

Console

To retrieve the public DNS name of the primary node with the new console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then select the cluster where you want to retrieve the public DNS name.
3. Note the **Primary node public DNS** value in the **Summary** section of the cluster details page.

CLI

To retrieve the public DNS name of the primary node with 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 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 primary 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. For more information about accessing your key pair, see Amazon EC2 key pairs in the *Amazon EC2 User Guide*.

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 primary 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 primary 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 to Amazon EMR: Authorize inbound traffic.
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 primary 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 primary node, type the following command. Replace *ec2-###-##-##-###.compute-1.amazonaws.com* with the primary 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 primary 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 primary node, type the following command to close the SSH connection.

```
exit
```

If you're experiencing difficulty with using SSH to connect to your primary node, see Troubleshoot connecting to your instance.

Connect to the primary node using SSH on Windows

Windows users can use an SSH client such as PuTTY to connect to the primary node. Before connecting to the Amazon EMR primary 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 primary node using PuTTY.

For more information about converting your key, see Converting your private key using PuTTYgen in the *Amazon EC2 User Guide*.

To connect to the primary node using PuTTY

1. Ensure you've allowed inbound SSH traffic. For instructions, see Before you connect to Amazon EMR: Authorize inbound traffic.
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 primary node, type `hadoop` if you are prompted for a user name .

8. When you are done working on the primary 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 primary node, see Troubleshoot connecting to your instance.

Connect to the primary node using the AWS CLI

You can create an SSH connection with the primary 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 primary 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.

To connect to the primary node using the AWS CLI

1. Ensure you've allowed inbound SSH traffic. For instructions, see Before you connect to Amazon EMR: Authorize inbound traffic.
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 primary 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 primary 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 primary 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

Component	Service description	Service running by default	Port	Configuration key
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
	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.port
Oozie		Yes	11000	

Component	Service description	Service running by default	Port	Configuration key
Tez	Tez UI	Yes	8080	
YARN	Shuffle	Yes	13562	mapreduce.shuffle.port
	Localizer RPC	Yes	8040	yarn.nodemanager.localizer.address
		Yes	8041	
	NM Webapp address	Yes	8042	yarn.nodemanager.webapp.address
	RM web application	Yes	8088	yarn.resourcemanager.webapp.address
		Yes	8025	
	Scheduler	Yes	8030	yarn.resourcemanager.scheduler.address
	applications manager interface	Yes	8032	yarn.resourcemanager.address
	RM admin interface	Yes	8033	yarn.resourcemanager.admin.address

Component	Service description	Service running by default	Port	Configuration key
	JobHistory Server Web UI	Yes	19888	mapreduce.jobhistory.webapp.address
	JobHistory Server Admin Web UI	Yes	10033	mapreduce.jobhistory.admin.address
	JobHistory Server (RPC)	Yes	10020	mapreduce.jobhistory.address
	Application Timeline Server (RPC)	Yes	10200	yarn.timeline-service.address
	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 for your Amazon EMR cluster.

Hadoop and other applications that you install on your EMR cluster publish user interfaces as web sites that are hosted on the primary node. For security reasons, when using Amazon EMR Managed Security Groups, these web sites are only available on the primary node's local web server. For that reason, you need to connect to the primary node to view the web interfaces. For more information, see Connect to the Amazon EMR cluster primary node using SSH. 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 Amazon 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 Amazon 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/

Name of interface	URI
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/
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 primary node that are not available on the core and task nodes, the instructions in this document are specific to the Amazon EMR primary 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 primary node.

There are several ways you can access the web interfaces on the primary node. The easiest and quickest method is to use SSH to connect to the primary 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 primary node using SSH).

```
lynx http://ip-###-##-##-###.us-west-2.compute.internal:8088/
```

There are two remaining options for accessing web interfaces on the primary 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 primary node, configure SSH tunneling with local port forwarding, and use an Internet browser to open web interfaces hosted on the primary 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 primary 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 primary 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 Amazon EMR cluster primary node.

> **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 Amazon EMR primary node using local port forwarding
- Option 2, part 1: Set up an SSH tunnel to the primary node using dynamic port forwarding
- Option 2, part 2: Configure proxy settings to view websites hosted on the Amazon EMR cluster primary node

Option 1: Set up an SSH tunnel to the Amazon EMR primary node using local port forwarding

To connect to the local web server on the primary node, you create an SSH tunnel between your computer and the primary 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 primary 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 primary node's local web server.

Setting up an SSH tunnel using local port forwarding requires the public DNS name of the primary node and your key pair private key file. For information about how to locate the master public DNS name, see Retrieve the public DNS name of the primary node. For more information about accessing your key pair, see Amazon EC2 key pairs in the *Amazon EC2 User Guide*. For more information about the sites you might want to view on the primary node, see View web interfaces hosted on Amazon EMR clusters.

Set up an SSH tunnel to the primary 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 to Amazon EMR: Authorize inbound traffic.
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 ran 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 primary node, close the terminal windows.

Option 2, part 1: Set up an SSH tunnel to the primary node using dynamic port forwarding

To connect to the local web server on the primary node, you create an SSH tunnel between your computer and the primary 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 primary 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 primary 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 primary node, and those on the Internet.

Before you begin, you need the public DNS name of the primary node and your key pair private key file. For information about how to locate the primary public DNS name, see Retrieve the public DNS name of the primary node. For more information about accessing your key pair, see Amazon EC2 key pairs in the *Amazon EC2 User Guide*. For more information about the sites you might want to view on the primary node, see View web interfaces hosted on Amazon EMR clusters.

Set up an SSH tunnel to the primary 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 to Amazon EMR: Authorize inbound traffic.
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 *ec2-###-##-##-###.compute-1.amazonaws.com* with the primary 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 primary 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 Amazon EMR cluster primary node.
5. When you are done working with the web interfaces on the primary 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 primary 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. 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 to Amazon EMR: Authorize inbound traffic.
2. Create an SSH connection with the primary node as shown in Connect to the primary node using the AWS CLI.
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 primary 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 Amazon EMR cluster primary node.
6. When you are done working with the web interfaces on the primary 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 primary node using PuTTY

Windows users can use an SSH client such as PuTTY to create an SSH tunnel to the primary node. Before connecting to the Amazon EMR primary 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 primary node using PuTTY.

For more information about converting your key, see Converting your private key using PuTTYgen in the *Amazon EC2 User Guide*.

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 to Amazon EMR: Authorize inbound traffic.
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 primary 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 primary 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 primary 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 Amazon EMR cluster primary node.
14. When you are done working with the web interfaces on the primary node, close the PuTTY window.

Option 2, part 2: Configure proxy settings to view websites hosted on the Amazon EMR cluster primary 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 primary 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 primary 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 primary node using dynamic port forwarding. For more information about the available web interfaces, see View web interfaces hosted on Amazon EMR clusters.

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 primary node in Option 2, part 1: Set up an SSH tunnel to the primary node using dynamic port forwarding. 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*.*compute*.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*.*compute*.amazonaws.com*",
        "active": true,
        "pattern": "*ec2*.*compute*.amazonaws.com*",
        "importedPattern": "*ec2*.*compute*.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*.*compute*.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 primary 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 primary node. For a complete list of web interface URLs, see View web interfaces hosted on Amazon EMR clusters.

Submit work to an Amazon EMR cluster

This section describes the methods that you can use to submit work to an Amazon EMR cluster. To submit work, you can add steps, or you can interactively submit Hadoop jobs to the primary node.

Consider the following rules of step behavior when you submit steps to a cluster:

- A step ID can contain up to 256 characters.
- You can have up to 256 PENDING and RUNNING steps in a cluster.
- Even if you have 256 active steps running on a cluster, you can interactively submit jobs to the primary node. 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. For more information, see Cancel steps when you submit work to an Amazon EMR cluster.
- 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. For more information, see Considerations for running multiple steps in parallel when you submit work to Amazon EMR.

> **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 with the Amazon EMR Management Console
- Adding steps to an Amazon EMR cluster with the AWS CLI
- Considerations for running multiple steps in parallel when you submit work to Amazon EMR
- Viewing steps after submitting work to an Amazon EMR cluster
- Cancel steps when you submit work to an Amazon EMR cluster

Adding steps to a cluster with the Amazon EMR Management Console

Use the following procedures to add steps to a cluster with the AWS Management Console. For detailed information about how to submit steps for specific big data applications, see the following sections of the *Amazon EMR Release Guide*:

- Submit a custom JAR step
- Submit a Hadoop streaming step
- Submit a Spark step
- Submit a Pig step
- Run a command or script as a step
- Pass values into steps to run Hive scripts

Add steps during cluster creation

From the AWS Management Console, you can add steps when you create a cluster.

Console

To add steps when you create a cluster with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose **Create cluster**.
3. Under **Steps**, choose **Add step**. Enter appropriate values in the fields in the **Add step** dialog. For information on formatting your step arguments, see Add step arguments. Options differ depending on the step type. To add your step and exit the dialog, select **Add step**.
4. Choose any other options that apply to your cluster.
5. To launch your cluster, choose **Create cluster**.

Add steps to a running cluster

With the AWS Management Console, you can add steps to a cluster with the auto-terminate option disabled.

Console

To add steps to a running cluster with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and select the cluster that you want to update.
3. On the **Steps** tab on the cluster details page, select **Add step**. To clone an existing step, choose the **Actions** dropdown menu and select **Clone step**.
4. Enter appropriate values in the fields in the **Add step** dialog. Options differ depending on the step type. To add your step and exit the dialog, choose **Add step**.

Modify the step concurrency level in a running cluster

With the AWS Management Console, you can modify the step concurrency level in a running cluster.

> **Note**
>
> You can only run multiple steps in parallel with Amazon EMR version 5.28.0 and later.

Console

To modify step concurrency in a running cluster with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and select the cluster that you want to update. The cluster must be running to change its concurrency attribute.
3. On the **Steps** tab on the cluster details page, find the **Attributes** section. Select **Edit** to change the concurrency. Enter a value between 1 and 256.

Add step arguments

When you use the AWS Management Console to add a step to your cluster, 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://amzn-s3-demo-bucket/my-script.sh ."
```

You can also put each argument on a separate line for readability as shown in the following example.

```
bash
-c
"aws s3 cp s3://amzn-s3-demo-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://amzn-s3-demo-bucket/my-script.sh .`, contains whitespace and is not surrounded by quotation marks.

```
bash -c aws s3 cp s3://amzn-s3-demo-bucket/my-script.sh .
```

Adding steps to an Amazon EMR cluster with the AWS CLI

The following procedures demonstrate how to add steps to a newly created cluster and to a running cluster with the AWS CLI. Both examples use the `--steps` subcommand 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=PRIMARY,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 with 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` with the `ModifyCluster` API. For example, type the following command to increase the `StepConcurrencyLevel` 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 when you submit work to Amazon EMR

Running multiple steps in parallel when you submit work to Amazon EMR requires preliminary decisions about resource planning and expectations regarding cluster behavior. These are covered in detail here.

- 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 primary node instance type meets the memory requirements of user workloads. The main step executer process runs on the primary node for each step. Running multiple steps in parallel requires more memory and CPU utilization from the primary 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 after submitting work to an Amazon EMR cluster

You can see up to 10,000 steps that Amazon EMR completed within the last seven days. You can also view 1,000 steps that Amazon EMR completed any time. This total includes both user-submitted and system steps.

If you submit new steps once the cluster reaches the 1,000 step record limit, Amazon EMR deletes the inactive user-submitted steps whose statuses have been COMPLETED, CANCELLED, or FAILED for more than seven days. If you submit steps beyond the 10,000 step record limit, Amazon EMR deletes the inactive user-submitted step records regardless of their inactive duration. Amazon EMR doesn't remove these records from the log files. Amazon EMR removes them from the AWS console, and they aren't returned 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	Up to 10,000 steps
Amazon EMR CLI	256 steps	NA
AWS CLI	NA	Up to 10,000 steps
API	256 steps	Up to 10,000 steps

Cancel steps when you submit work to an Amazon EMR cluster

You can cancel pending and running steps from the AWS Management Console, the AWS CLI, or the Amazon EMR, when you submit work to your cluster. API.

Console

To cancel steps with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then select the cluster that you want to update.
3. On the **Steps** tab on the cluster details page, select the check box next to the step you wants to cancel. Choose the **Actions** dropdown menu and then select **Cancel steps**.
4. In the **Cancel the step** dialog, choose to either cancel the step and wait for it to exit, or cancel the step and force it to exit. Then choose **Confirm**.
5. The status of the steps in the **Steps** table changes to `CANCELLED`.

CLI

To cancel with 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.

View and monitor an Amazon EMR cluster as it performs work

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 primary 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 Amazon EMR cluster status and details
- Enhanced step debugging with Amazon EMR
- View Amazon EMR application history
- View Amazon EMR log files
- View cluster instances in Amazon EC2
- CloudWatch events and metrics from Amazon EMR

- View cluster application metrics using Ganglia with Amazon EMR
- Logging AWS EMR API calls using AWS CloudTrail
- EMR Observability Best Practices

View Amazon EMR 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.

View cluster details using the AWS Management Console

The **Clusters** list in the https://console.aws.amazon.com/emr 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** and **Status details**, 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.

Console

To view cluster information with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and select the cluster that you want to view.

3. Use the **Summary** panel to view the basics of your cluster configuration, such as cluster status, the open-source applications that Amazon EMR installed on the cluster, and the version of Amazon EMR that you used to create the cluster. Use each tab below the Summary to view information as described in the following table.

View cluster details 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. 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.

Enhanced step debugging with Amazon EMR

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.

Console

The new Amazon EMR console doesn't offer step debugging. However, you can view cluster termination details with the following steps.

To view failure details with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then select the cluster that you want to view.
3. Note the **Status** value in the **Summary** section of the cluster details page. If the status is **Terminated with errors**, hover over the text to view cluster failure details.

CLI

To view failure details with the AWS CLI

- To get failure details for a step with 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://amzn-s3-demo-bucket/logs/j-1K48XXXXXHCB/steps/
s-3QM0XXXXXM1W/stderr.gz",
        "Message": "org.apache.hadoop.mapred.FileAlreadyExistsException: Output
 directory s3://amzn-s3-demo-bucket/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://amzn-s3-demo-bucket/input/input.txt",
        "s3://amzn-s3-demo-bucket/logs/beta"
      ],
      "Jar": "s3://amzn-s3-demo-bucket/jars/hadoop-mapreduce-examples-2.7.2-
amzn-1.jar",
      "Properties": {}
    },
    "Id": "s-3QM0XXXXXM1W",
    "ActionOnFailure": "CONTINUE",
    "Name": "ExampleJob"
  }
```

```
}
```

View Amazon EMR application history

You can view Spark History Server and YARN timeline service application details with the 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.

> **Note**
>
> To augment the security for the off-console applications that you might use with Amazon EMR, the application hosting domains are registered in the Public Suffix List (PSL). Examples of these hosting domains include the following: `emrstudio-prod.us-east-1.amazonaws.com`, `emrnotebooks-prod.us-east-1.amazonaws.com`, `emrappui-prod.us-east-1.amazonaws.com`. For further security, if you ever need to set sensitive cookies in the default domain name, we recommend that you use cookies with a `__Host-` prefix. This helps to defend your domain against cross-site request forgery attempts (CSRF). For more information, see the Set-Cookie page in the *Mozilla Developer Network*.

The **Application user interfaces** section of the **Applications** tab provides several viewing options, depending on the cluster status and the applications you installed on the cluster.

- 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 UI and Spark History Service. 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.

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.

View persistent application user interfaces in Amazon EMR

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.

Navigate to your cluster details in the console, and select the **Applications** tab. Select the application UI that you want once your cluster has launched. 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 you don't enable logging, the links to YARN container logs won't 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 the correct system bucket ARNs 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-7.10.0
 \
--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
```

Permissions for viewing persistent application user interfaces

The following sample shows the role permissions required for access to persistent application user interfaces. For clusters with runtime role enabled, this will only allow users to access applications submitted by the same user identity and runtime role.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:CreatePersistentAppUI",
        "elasticmapreduce:DescribePersistentAppUI"
      ],
      "Resource": [
        "arn:aws:elasticmapreduce:*:123456789012:cluster/clusterId"
      ],
      "Sid": "AllowELASTICMAPREDUCECreatepersistentappui"
    },
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:GetPersistentAppUIPresignedURL"
      ],
      "Resource": [
        "arn:aws:elasticmapreduce:*:123456789012:cluster/clusterId",
        "arn:aws:elasticmapreduce:*:123456789012:persistent-app-ui/*"
      ],
      "Condition": {
        "StringEqualsIfExists": {
          "elasticmapreduce:ExecutionRoleArn": [
            "arn:aws:iam::123456789012:role/executionRoleArn"
          ]
        }
      },
      "Sid": "AllowELASTICMAPREDUCEGetpersistentappuipresignedurl"
    }
  ]
}
```

The following sample shows the role permissions required for removing the restrictions on viewing applications in the persistent application user interfaces for runtime role enabled clusters.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:CreatePersistentAppUI",
        "elasticmapreduce:DescribePersistentAppUI",
        "elasticmapreduce:AccessAllEventLogs"
      ],
      "Resource": [
        "arn:aws:elasticmapreduce:us-east-1:123456789012:cluster/j-XXXXXXXXXXXXX"
      ],
      "Sid": "AllowELASTICMAPREDUCECreatepersistentappui"
    },
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:GetPersistentAppUIPresignedURL"
      ],
      "Resource": [
        "arn:aws:elasticmapreduce:us-east-1:123456789012:cluster/j-
XXXXXXXXXXXXX",
        "arn:aws:elasticmapreduce:us-east-1:123456789012:persistent-app-ui/*"
      ],
      "Condition": {
        "StringEqualsIfExists": {
          "elasticmapreduce:ExecutionRoleArn": [
            "arn:aws:iam::123456789012:role/YourExecutionRoleName"
          ]
        }
      },
      "Sid": "AllowELASTICMAPREDUCEGetpersistentappuipresigneduurl"
    }
  ]
}
```

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 `CreatePersistentAppUI`, `DescribePersistentAppUI` and `GetPersistentAppUIPresignedURL` actions for Amazon EMR. If you deny an IAM principal's permission to these actions, it takes approximately five minutes for the permission change to propagate.
- If a cluster is a runtime role enabled cluster, when accessing the Spark History Server from the Persistent App UI, the user will only be able to access a Spark job if the Spark job is submitted by a runtime role.
- If a cluster is a runtime role enabled cluster, each user can access only an application submitted by the same user identity and runtime role.
- The `AccessAllEventLogs` action for Amazon EMR is necessary to view all applications in persistent application user interfaces for runtime role enabled clusters.
- 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.
- In the following AWS Regions, you can access application UIs from the console with Amazon EMR 6.14.0 and higher:
  - Asia Pacific (Jakarta) (ap-southeast-3)
  - Europe (Spain) (eu-south-2)
  - Asia Pacific (Melbourne) (ap-southeast-4)
  - Israel (Tel Aviv) (il-central-1)
  - Middle East (UAE) (me-central-1)

- In the following AWS Regions, you can access application UIs from the console with Amazon EMR 5.25.0 and higher:
  - US East (N. Virginia) (us-east-1)
  - US West (Oregon) (us-west-2)
  - Asia Pacific (Mumbai) (ap-south-1)
  - 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)
  - South America (São Paulo) (sa-east-1)
  - Europe (Frankfurt) (eu-central-1)
  - Europe (Ireland) (eu-west-1)
  - Europe (London) (eu-west-2)
  - Europe (Paris) (eu-west-3)
  - Europe (Stockholm) (eu-north-1)
  - China (Beijing) (cn-north-1)
  - China (Ningxia) (cn-northwest-1)

View a high-level application history in Amazon EMR

> **Note**
>
> We recommend that you use the persistent application interface for an improved user experience that retains app history for up to 30 days. The high-level application history described on this page isn't available in the new Amazon EMR console (https://console.aws.amazon.com/emr). For more information, see View persistent application user interfaces in Amazon EMR.

With Amazon EMR releases 5.8.0 to 5.36.0 and 6.x releases up to 6.8.0, you can view a high-level application history from the **Application user interfaces** tab in the old Amazon EMR console. An

Amazon EMR **Application user interface** keeps the summary of application history for 7 days after an application has completed.

Considerations and limitations

Consider the following limitations when you use the **Application user interfaces** tab in the old 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. Effective January 23, 2023, Amazon EMR will discontinue high-level application history for all versions. If you use Amazon EMR version 5.25.0 or higher, we recommend that you use the persistent application user interface instead.
- 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 of the old console.

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.

Cluster: Development Cluster Waiting Cluster ready to run steps.

Summary | Application user interfaces | Monitoring | Hardware | Configurations | Events | Steps | Bootstrap actions

Persistent application user interfaces

Applications installed on the Amazon EMR cluster publish user interfaces (UI) as web sites to monitor cluster activity. Persistent UI logs are available for 30 days after an application ends. Persistent UI don't required SSH tunneling. They are hosted off of the cluster.

Application user interface
YARN timeline server
Tez UI
Spark history server

On-cluster application user interfaces

On-cluster UI are available only while clusters are running. Because they are hosted on the master node, on-cluster UI require a connection via SSH tunneling. Set up SSH tunneling before accessing these application UI. Learn more

Application	User interface URL	Status
Spark History Server	http:// compute-1.amazonaws.com:18080/	SSH tunnel not enabled

High-level application history

Amazon EMR collects information from YARN applications on your cluster and keeps a summary of historical information for seven days after applications have completed. Learn more

YARN applications (5)

Filter: All applications | Filter applications ... | 5 applications (all loaded)

Application ID	Type	Action	Status	Start time (UTC-7)	Duration	Finish time (UTC-7)	User
application_1590503538546_0005	TEZ	HIVE-62d52467-d2ac-4430-98b9-9859317f5673	Succeeded	2020-05-26 07:56 (UTC-7)	5.2 min	2020-05-26 08:02 (UTC-7)	hadoop
application_1590503538546_0004	TEZ	HIVE-ea51ce39-4c0f-44f9-9613-bc8037f07710	Succeeded	2020-05-26 07:56 (UTC-7)	5.2 min	2020-05-26 08:02 (UTC-7)	hadoop
application_1590503538546_0003	Spark	Spark shell	Succeeded	2020-05-26 07:50 (UTC-7)	5.5 min	2020-05-26 07:56 (UTC-7)	hadoop
Diagnostics: Succeeded							
application_1590503538546_0002	Spark	Spark shell	Succeeded	2020-05-26 07:47 (UTC-7)	2.1 min	2020-05-26 07:49 (UTC-7)	hadoop
application_1590503538546_0001	TEZ	HIVE-a5e557a7-dfbe-4577-87ed-4326eb7cc0f3	Succeeded	2020-05-26 07:33 (UTC-7)	5.2 min	2020-05-26 07:38 (UTC-7)	hive

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.

Cluster: Development Cluster Waiting Cluster ready to run steps.

Summary | Application user interfaces | Monitoring | Hardware | Configurations | Events | Steps | Bootstrap actions

Persistent application user interfaces

Applications installed on the Amazon EMR cluster publish user interfaces (UI) as web sites to monitor cluster activity. Persistent UI logs are available for 30 days after an application ends. Persistent UI don't required SSH tunneling. They are hosted off of the cluster.

Application user interface
YARN timeline server
Tez UI
Spark history server

On-cluster application user interfaces

On-cluster UI are available only while clusters are running. Because they are hosted on the master node, on-cluster UI require a connection via SSH tunneling. Set up SSH tunneling before accessing these application UI. Learn more

Application	User interface URL	Status
Spark History Server	http://[redacted].compute-1.amazonaws.com:18080/	SSH tunnel not enabled

High-level application history

YARN applications > application_1590503538546_0003 (Spark)

Jobs | Stages | Executors

User: hadoop
Total uptime: 5.6 min
Completed jobs: 10

Event timeline

Jobs (10)

Filter: Filter jobs ... 10 jobs (all loaded)

Job ID	Status	Description	Submitted (UTC-7)	Duration	Stages succeeded / total	Tasks succeeded / total
9	Succeeded	collect at HoodieCopyOnWriteTable.java:329	2020-05-26 07:52 (UTC-7)	82 ms	2 / 2	4 / 4
8	Succeeded	collect at HoodieCopyOnWriteTable.java:304	2020-05-26 07:52 (UTC-7)	1 s	1 / 1	2 / 2
7	Succeeded	collect at AbstractHoodieWriteClient.java:140	2020-05-26 07:52 (UTC-7)	63 ms	1 / 6	1 / 4,503
6	Succeeded	count at HoodieSparkSqlWriter.scala:257	2020-05-26 07:52 (UTC-7)	6 s	2 / 6	1,501 / 4,503
5	Succeeded	countByKey at WorkloadProfile.java:67	2020-05-26 07:52 (UTC-7)	9 s	5 / 6	6,001 / 6,002
4	Succeeded	countByKey at HoodieBloomIndex.java:174	2020-05-26 07:52 (UTC-7)	4 s	2 / 3	3,000 / 3,001
3	Succeeded	collect at HoodieBloomIndex.java:218	2020-05-26 07:52 (UTC-7)	3 s	1 / 1	1 / 1
2	Succeeded	collect at HoodieBloomIndex.java:205	2020-05-26 07:52 (UTC-7)	3 s	1 / 1	1 / 1
1	Succeeded	countByKey at HoodieBloomIndex.java:141	2020-05-26 07:52 (UTC-7)	7 s	3 / 3	3,001 / 3,001
0	Succeeded	isEmpty at HoodieSparkSqlWriter.scala:142	2020-05-26 07:52 (UTC-7)	8 s	1 / 1	1 / 1

On the job details page, you can expand information about individual job stages, and then select the **Description** link to see stage details.

Cluster: Development Cluster Waiting Cluster ready to run steps.

Summary | Application user interfaces | Monitoring | Hardware | Configurations | Events | Steps | Bootstrap actions

Persistent application user interfaces

Applications installed on the Amazon EMR cluster publish user interfaces (UI) as web sites to monitor cluster activity. Persistent UI logs are available for 30 days after an application ends. Persistent UI don't required SSH tunneling. They are hosted off of the cluster.

Application user interface
YARN timeline server
Tez UI
Spark history server

On-cluster application user interfaces

On-cluster UI are available only while clusters are running. Because they are hosted on the master node, on-cluster UI require a connection via SSH tunneling. Set up SSH tunneling before accessing these application UI. Learn more

Application	User interface URL	Status
Spark History Server	http://[redacted]compute-1.amazonaws.com:18080/	SSH tunnel not enabled

High-level application history

YARN applications > application_1590503538546_0003 (Spark)

Jobs | Stages | Executors

Jobs > Job 9

Status: Succeeded

Completed stages: 2

Event timeline

Stages (2)

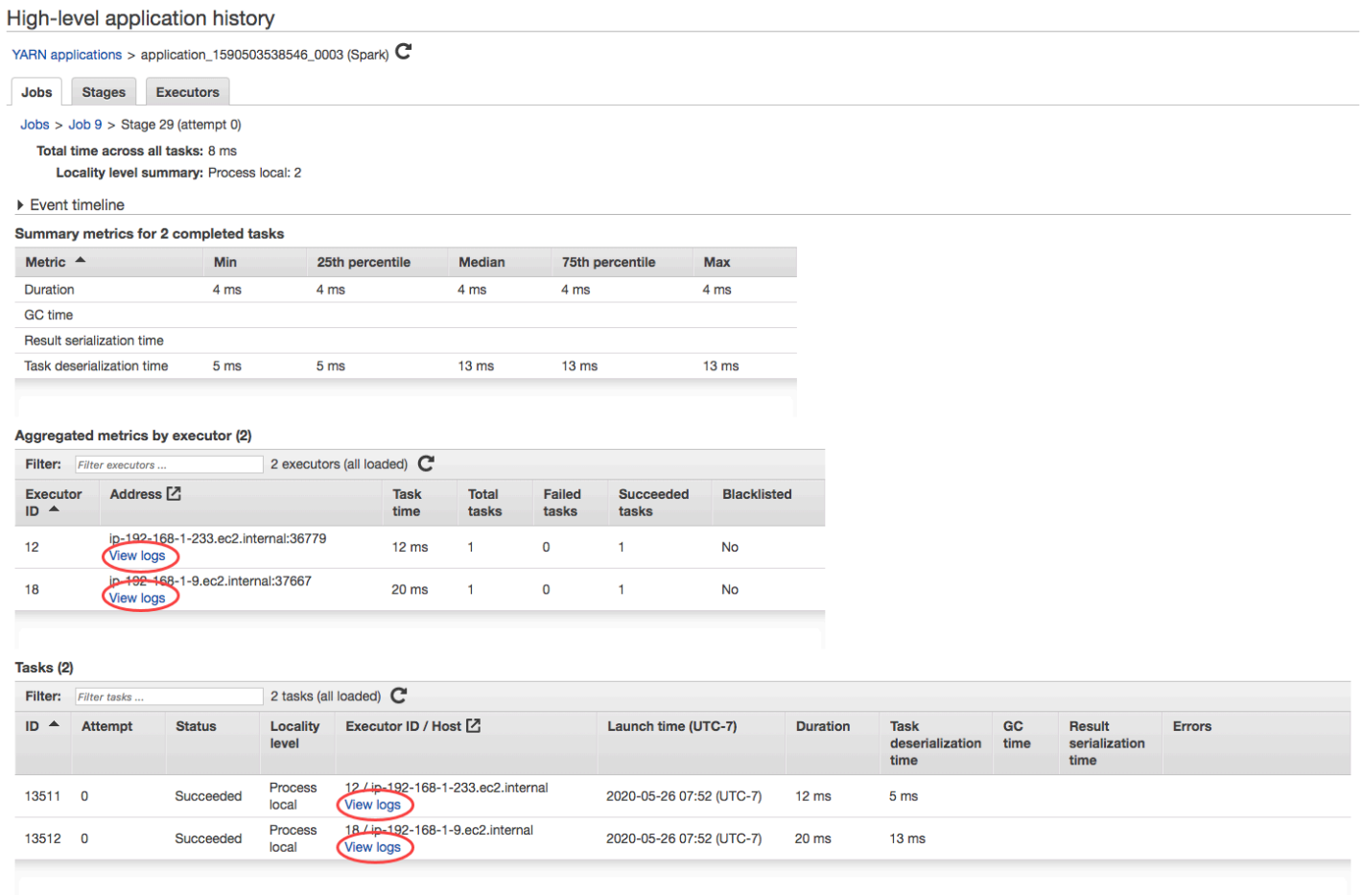
Filter: Filter stages ... 2 stages (all loaded)

Stage ID	Status	Description	Submitted (UTC-7)	Duration	Tasks succeeded / total	Input	Output	Shuffle read	Shuffle write
29	Completed	collect at HoodieCopyOnWriteTable.java:329	2020-05-26 07:52 (UTC-7)	20 ms	2 / 2				
28	Completed	mapPartitionsToPair at HoodieCopyOnWriteTable.java:329	2020-05-26 07:52 (UTC-7)	31 ms	2 / 2				

Details:

```
org.apache.spark.api.java.AbstractJavaRDDLike.collect(JavaRDDLike.scala:45)
org.apache.hudi.table.HoodieCopyOnWriteTable.clean(HoodieCopyOnWriteTable.java:329)
org.apache.hudi.client.HoodieCleanClient.runClean(HoodieCleanClient.java:163)
org.apache.hudi.client.HoodieCleanClient.clean(HoodieCleanClient.java:98)
org.apache.hudi.client.HoodieWriteClient.clean(HoodieWriteClient.java:835)
org.apache.hudi.client.HoodieWriteClient.postCommit(HoodieWriteClient.java:512)
org.apache.hudi.client.AbstractHoodieWriteClient.commit(AbstractHoodieWriteClient.java:157)
org.apache.hudi.client.AbstractHoodieWriteClient.commit(AbstractHoodieWriteClient.java:101)
org.apache.hudi.client.AbstractHoodieWriteClient.commit(AbstractHoodieWriteClient.java:92)
org.apache.hudi.HoodieSparkSqlWriter$.checkWriteStatus(HoodieSparkSqlWriter.scala:263)
org.apache.hudi.HoodieSparkSqlWriter$.write(HoodieSparkSqlWriter.scala:184)
org.apache.hudi.DefaultSource.createRelation(DefaultSource.scala:91)
org.apache.spark.sql.execution.datasources.SaveIntoDataSourceCommand.run(SaveIntoDataSourceCommand.scala:46)
org.apache.spark.sql.execution.command.ExecutedCommandExec.sideEffectResult$lzycompute(commands.scala:70)
org.apache.spark.sql.execution.command.ExecutedCommandExec.sideEffectResult(commands.scala:68)
org.apache.spark.sql.execution.command.ExecutedCommandExec.doExecute(commands.scala:86)
org.apache.spark.sql.execution.SparkPlan.$anonfun$execute$1(SparkPlan.scala:131)
org.apache.spark.sql.execution.SparkPlan.$anonfun$executeQuery$1(SparkPlan.scala:156)
org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:151)
org.apache.spark.sql.execution.SparkPlan.executeQuery(SparkPlan.scala:152)
```

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 Amazon EMR log files

Amazon EMR and Hadoop both produce log files that report status on the cluster. By default, these are written to the primary 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 primary 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.

View log files on the primary node

The following table lists some of the log files you'll find on the primary 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.
/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.

Location	Description
	• **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 primary node with the AWS CLI.

1. Use SSH to connect to the primary node as described in Connect to the Amazon EMR cluster primary node using SSH.
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.

Location	Description
	• **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 with the Amazon S3 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 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 that you assign to resources through your AWS account are available only to you. Other accounts that share the same resource can't view your tags.

Amazon EMR automatically tags each EC2 instance that it launches with key-value pairs. The keys 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 that belong to a particular

cluster, or to show all of the currently running instances in the instance group for the task. This is especially useful if you run several clusters concurrently or manage large numbers of EC2 instances.

These are the predefined key-value pairs that Amazon EMR assigns:

Key	Value	Value definition
aws:elasticmapreduce:job-flow-id	*job-flow-identifier*	The ID of the cluster that the instance is provisioned for. It appears in the format `j-XXXXXXXXXXXXX` and can be up to 256 characters long.
aws:elasticmapreduce:instance-group-role	*group-role*	The type of instance group, entered as one of the following values: `master`, `core`, or `task`.

You can view and filter on the tags that Amazon EMR adds. For more information, see Using tags in the *Amazon EC2 User Guide*. 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 updates to **Running**. If latency occurs between the time that the EC2 instance is provisioned and the time that its status is set to **Running**, the tags that Amazon EMR sets will appear once the instance starts. If you don't see the tags, wait for a few minutes and refresh the view.

CloudWatch events and metrics from Amazon EMR

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 to monitor 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

- Monitoring Amazon EMR metrics with CloudWatch
- Monitoring Amazon EMR events with CloudWatch
- Responding to CloudWatch events from Amazon EMR

Monitoring Amazon EMR 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.

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.

How do I?	Relevant metrics
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 Amazon EMR cluster terminates with NO_SLAVE_LEFT and core nodes FAILED_BY_MASTER 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.

Console

To view metrics with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose the cluster that you want to view metrics for. This opens the cluster details page.
3. Select the **Monitoring** tab on the cluster details page. Choose any one of the **Cluster status**, **Node status**, or **Inputs and outputs** options to load the reports about the progress and health of the cluster.

4. After you choose a metric to view, you can enlarge each graph. To filter the time frame of your graph, select a prefilled option or choose **Custom**.

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.

Metric	Description
	Use case: Monitor cluster performance 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*

Metric	Description
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*
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*

Metric	Description
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 correspon ding 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*

Metric	Description
MRTotalNodes	The number of nodes presently available to MapReduce jobs. Equivalent to YARN metric `mapred.resourceman ager.TotalNodes` . Use ase: Monitor cluster progress Units: *Count* Note: MRTotalNodes only counts currently active nodes in the system. YARN automatically removes terminated nodes from this count and stops tracking them, so they are not considered in the MRTotalNodes metric.
MRActiveNodes	The number of nodes presently running MapReduce tasks or jobs. Equivalent to YARN metric `mapred.re sourcemanager.NoOfActiveNodes` . Use case: Monitor cluster progress Units: *Count*
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.NoOfUnhealthy Nodes` . Use case: Monitor cluster progress Units: *Count*

Metric	Description
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 Amazon EMR. Use case: Analyze cluster performance, Monitor cluster progress Units: *Count*

Metric	Description
HDFSBytesWritten	The number of bytes written to HDFS. 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*
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*
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*

Metric	Description
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

Metric	Description
	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*
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*

Metric	Description
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*

Metric	Description
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 correspon ding 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*
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 correspon ding instance group exists. Use case: Monitor cluster health Units: *Count*

Metric	Description
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*
HDFSBytesRead	The number of bytes read from HDFS. Use case: Analyze cluster performance, Monitor cluster progress Units: *Count*

Metric	Description
HDFSBytesWritten	The number of bytes written to HDFS. Use case: Analyze cluster performance, Monitor cluster p rogress 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 contribut ing 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. Units: *Count*
• `CoreUnitsRequested` • `CoreNodesRequested` • `CoreVCPURequested`	The target number of CORE units/nodes/vCPUs in a cluste r 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*

Metric	Description
• `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 for Amazon EMR cluster cleanup.

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

Metric	Description
	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.

Monitoring Amazon EMR events with CloudWatch

Amazon EMR tracks events and keeps information about them for up to seven days in the Amazon EMR console. Amazon EMR records events when there is a change in the state of clusters, instance groups, instance fleets, automatic scaling policies, or steps. Events capture the date and time the event occurred, details about the affected elements, and other critical data points.

The following table lists Amazon EMR events, along with the state or state change that the event indicates, the severity of the event, event type, event code, and event messages. Amazon EMR represents events as JSON objects and automatically sends them to an event stream. The JSON object is 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*.

> **Note**
>
> EMR periodically emits events with the event code **EC2 provisioning - Insufficient Instance Capacity**. These events occur when your Amazon EMR cluster encounters an insufficient capacity error from Amazon EMR for your instance fleet or instance group during cluster creation or resize operation. An event might not include all the instance types and AZs you have provided, because EMR only includes the instance types and AZs it attempted to provision capacity in since the last the Insufficient capacity event was emitted. For information on how to respond to these events, see Responding to Amazon EMR cluster insufficient instance capacity events.

Cluster start events

State or state change	Severity	Event type	Event code	Message
`CREATING`	`WARN`	EMR instance fleet provisioning	EC2 provisioning - Insufficient Instance Capacity	We are not able to create your Amazon EMR cluster `ClusterId (ClusterName)` for Instance Fleet `InstanceFleetID` Amazon EC2 has insufficient Spot capacity for Instance type `[Instancetype1, Instancetype2]` and insufficient On-Demand

State or state change	Severity	Event type	Event code	Message
				capacity for Instance type `[Instance type3, Instancetype4]` in Availability Zone `[AvailabilityZone1, AvaliabilityZone2]`. Check here documentation for more information on how to respond to this event.

State or state change	Severity	Event type	Event code	Message
CREATING	WARN	EMR instance group provisioning	EC2 provisioning - Insufficient Instance Capacity	We are not able to create your Amazon EMR cluster ClusterId (ClusterName) for Instance Group InstanceGroupID Amazon EC2 has insufficient Spot capacity for Instance type [Instancetype1, Instancetype2] and insufficient On-Demand capacity for Instance type [Instancetype3, Instancetype4] in Availability Zone [AvailabilityZone1, AvaliabilityZone2]. Check here

State or state change	Severity	Event type	Event code	Message
				documentation for more information on how to respond to this event.

State or state change	Severity	Event type	Event code	Message
`CREATING`	`WARN`	EMR instance fleet provisioning	EC2 provisioning - Insufficient Free Addresses In Subnet	We can’t create the Amazon EMR cluster `ClusterId (ClusterName)` that you requested for instance fleet `InstanceFleetID` because the specified subnet `[Subnet1, Subnet2]` doesn't contain enough free private IP addresses to fulfill your request. Use the `DescribeSubnets` operation to see how many IP addresses are available (unused) in your subnet. For information on how to respond to this event, see Error codes for

State or state change	Severity	Event type	Event code	Message
				the Amazon EC2 API

State or state change	Severity	Event type	Event code	Message
`CREATING`	`WARN`	EMR instance group provisioning	EC2 provisioning - Insufficient Free Addresses In Subnet	We can’t create the Amazon EMR cluster `ClusterId (ClusterName)` that you requested for instance group `InstanceGroupID` because the specified subnet `[Subnet1, Subnet2]` doesn't contain enough free private IP addresses to fulfill your request. Use the `DescribeSubnets` operation to see how many IP addresses are available (unused) in your subnet. For information on how to respond to this event, see Error codes for

State or state change	Severity	Event type	Event code	Message
				the Amazon EC2 API
`CREATING`	`WARN`	EMR instance fleet provisioning	EC2 Provisioning – vCPU Limit Exceeded	The provision of `InstanceFleetID` in the Amazon EMR cluster `ClusterId (ClusterName)` is delayed because you've reached the limit on the number of vCPUs (virtual processing units) assigned to the running instances in your `account (accountId)`. For more information, Error codes for the Amazon EC2 API

State or state change	Severity	Event type	Event code	Message
`CREATING`	`WARN`	EMR instance group provisioning	EC2 Provisioning – vCPU Limit Exceeded	The provision of instance group `InstanceGroupID` in the Amazon EMR cluster `ClusterId` is delayed because you've reached the limit on the number of vCPUs (virtual processing units) assigned to the running instances in your account `(accountId)`. For more information, Error codes for the Amazon EC2 API

State or state change	Severity	Event type	Event code	Message
`CREATING`	`WARN`	EMR instance fleet provisioning	EC2 Provisioning – Spot Instance Count Limit Exceeded	The provision of instance fleet `InstanceFleetID` in the Amazon EMR cluster `ClusterID (ClusterName)` is delayed because you've reached the limit on the number of Spot Instances that you can launch in your `account (accountId)`. For more information, see Error codes for the Amazon EC2 API.

State or state change	Severity	Event type	Event code	Message
`CREATING`	`WARN`	EMR instance group provisioning	EC2 Provisioning – Spot Instance Count Limit Exceeded	The provision of instance group `InstanceGroupID` in the Amazon EMR cluster `ClusterID (ClusterName)` is delayed because you've reached the limit on the number of Spot Instances that you can launch in your `account (accountId)`. For more information, see Error codes for the Amazon EC2 API.

State or state change	Severity	Event type	Event code	Message
`CREATING`	`WARN`	EMR instance fleet provisioning	EC2 Provisioning – Instance Limit Exceeded	The provision of instance fleet `InstanceFleetID` in the Amazon EMR cluster `ClusterId (ClusterName)` is delayed because you've reached the limit on the number of instances you can run concurrently in your `account (accountID)`. For more information on Amazon EC2 service limits, see Error codes for the Amazon EC2 API.

State or state change	Severity	Event type	Event code	Message
`CREATING`	`WARN`	EMR instance group provisioning	EC2 Provisioning – Instance Limit Exceeded	The provision of instance group `InstanceGroupID` in the Amazon EMR cluster `ClusterId` `(ClusterName)` is delayed because you've reached the limit on the number of instances you can run concurrently in your `account` `(accountID)`. For more information on Amazon EC2 service limits, see Error codes for the Amazon EC2 API.

State or state change	Severity	Event type	Event code	Message
`CREATING`	`WARN`	EMR instance group provisioning	*none*	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 still be processing jobs.

State or state change	Severity	Event type	Event code	Message
`STARTING`	`INFO`	EMR cluster state change	*none*	Amazon EMR cluster `ClusterId` `(ClusterName)` was requested at `Time` and is being created.

State or state change	Severity	Event type	Event code	Message
`STARTING`	`INFO`	EMR cluster state change	*none*	**Note** Applies only to clusters with the instance fleets configuration and multiple Availability Zones selected within Amazon EC2. Amazon EMR cluster `ClusterId` (`ClusterName`) is being created in zone (`AvailabilityZoneID`), which was chosen from the specified Availability Zone options.

State or state change	Severity	Event type	Event code	Message
`STARTING`	`INFO`	EMR cluster state change	*none*	Amazon EMR cluster `ClusterId` `(ClusterName)` began running steps at `Time`.

State or state change	Severity	Event type	Event code	Message
`WAITING`	`INFO`	EMR cluster state change	*none*	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 still be processing jobs.

> **Note**
>
> The events with event code `EC2 provisioning - Insufficient Instance Capacity` periodically emit when your EMR cluster encounters an insufficient capacity error from Amazon EC2 for your instance fleet or instance group during cluster creation or resize operation. For information on how to respond to these events, see Responding to Amazon EMR cluster insufficient instance capacity events.

Cluster termination events

State or state change	Severity	Event type	Event code	Message
`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`,	EMR cluster state change	*none*	Amazon EMR Cluster `ClusterId` (`ClusterName`) has terminated at `Time` with a reason of `StateChangeReason: Code` .

State or state change	Severity	Event type	Event code	Message
	or `STEP_FAILURE`. • **`INFO`** if the cluster terminated with any of the following state change reasons: `USER_REQUEST` or `ALL_STEPS_COMPLETED`.			
`TERMINATED_WITH_ERRORS`	`CRITICAL`	EMR cluster state change	*none*	Amazon EMR Cluster `ClusterId` `(ClusterName)` has terminated with errors at `Time` with a reason of `StateChangeReason: Code`.

State or state change	Severity	Event type	Event code	Message
`TERMINATED_WITH_ERRORS`	`CRITICAL`	EMR cluster state change	*none*	Amazon EMR Cluster `ClusterId` (`ClusterName`) has terminated with errors at `Time` with a reason of `StateChangeReason: Code` .

Instance fleet state-change 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	Event type	Event code	Message
From `PROVISIONING` to `WAITING`	`INFO`		*none*	Provisioning for instance fleet `InstanceFleetID` in Amazon EMR cluster `ClusterId` (`ClusterName`) is complete.

State or state change	Severity	Event type	Event code	Message
				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 w as `Num`.

State or state change	Severity	Event type	Event code	Message
From `WAITING` to `RESIZING`	`INFO`		*none*	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`.

State or state change	Severity	Event type	Event code	Message
From `RESIZING` to `WAITING`	`INFO`		*none*	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`.

State or state change	Severity	Event type	Event code	Message
From `RESIZING` to `WAITING`	`INFO`		*none*	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`.

State or state change	Severity	Event type	Event code	Message
SUSPENDED	ERROR		*none*	Instance fleet InstanceFleetID in Amazon EMR cluster ClusterId (ClusterName) was arrested at Time for the following reason: ReasonDesc .
RESIZING	WARNING		*none*	The resizing operation for instance fleet InstanceFleetID in Amazon EMR cluster ClusterId (ClusterName) is stuck for the following reason: ReasonDesc .

State or state change	Severity	Event type	Event code	Message
`WAITING` or `Running`	`INFO`		*none*	The resizing operation for instance fleet `InstanceFleetID` in Amazon EMR cluster `ClusterId` `(ClusterName)` couldn't complete while Amazon EMR added Spot capacity in availability zone `AvailabilityZone`. We've cancelled your request to provision additional Spot capacity. For recommended actions, check Availability Zone flexibility for an Amazon EMR cluster and try again.

State or state change	Severity	Event type	Event code	Message
`WAITING` or `Running`	`INFO`		*none*	A resizing operation for instance fleet `InstanceFleetID` in Amazon EMR cluster `ClusterId` `(ClusterName)` was initiated by `Entity` at `Time`.

Instance fleet reconfiguration events

State or state change	Severity	Message
Instance Fleet Reconfiguration Requested	`INFO`	A user has requested to reconfigure the instance fleet `InstanceFleetID` in Amazon EMR cluster `ClusterId` (`ClusterName`).
Instance Fleet Reconfiguration Start	`INFO`	Amazon EMR has started a reconfiguration of the instance fleet `InstanceFleetID` in the Amazon EMR cluster `ClusterId` (`ClusterName`) at `Time`.

State or state change	Severity	Message
Instance Fleet Reconfiguration Completed	INFO	Amazon EMR has finished reconfiguring instance fleet InstanceFleetID in the Amazon EMR cluster ClusterId (ClusterName).
Instance Fleet Reconfiguration Failed	WARNING	Amazon EMR failed to reconfigure the instance fleet InstanceFleetID in the Amazon EMR cluster ClusterId (ClusterName) at Time. The reconfiguration failed because Reason.
Instance Fleet Reconfiguration Reversion Start	INFO	Amazon EMR is reverting the instance fleet InstanceFleetID in the Amazon EMR cluster ClusterId (ClusterName) to the previous successful configuration.
Instance Fleet Reconfiguration Reversion Completed	INFO	Amazon EMR finished reverting the instance fleet InstanceFleetID in the Amazon EMR cluster ClusterId (ClusterName) to the previous successful configuration.

State or state change	Severity	Message
Instance Fleet Reconfiguration Reversion Failed	`CRITICAL`	Amazon EMR couldn't revert the instance fleet `InstanceFleetID` in the Amazon EMR cluster `ClusterId` (`ClusterName`) to the previously successful configuration at `Time`. The reconfiguration reversion failed because of `Reason`.
Instance Fleet Reconfiguration Reversion Blocked	`INFO`	Amazon EMR tmeporarily blocked the instance fleet `InstanceFleetID` in the Amazon EMR cluster `ClusterId` (`ClusterName`) at `Time` because the instance fleet is in the `State` state.

Instance fleet resize events

Event type	Severity	Event code	Message
EMR instance fleet resize	`ERROR`	Spot Provisioning timeout	The Resize operation for Instance Fleet `InstanceFleetID` in Amazon EMR cluster `ClusterId` (`ClusterName`) was not able to complete while acquiring Spot capacity in AZ `Availabil`

Event type	Severity	Event code	Message
			`ityZone` . We have now cancel led your request and stopped trying to provision any additi onal Spot capacity and the Instance Fleet has provision ed Spot capacity of `num`. Target Spot capacity was `num`. For more informati on and recommended actions, please check the documentation page here and retry again.

Event type	Severity	Event code	Message
EMR instance fleet resize	ERROR	On-Demand Provisioning timeout	The Resize operation for Instance Fleet `InstanceFleetID` in Amazon EMR cluster `ClusterId` `(ClusterName)` was not able to complete while acquiring On-Demand capacity in AZ `AvailabilityZone` . We have now cancelled your request and stopped trying to provision any additional On-Demand capacity and the Instance Fleet has provisioned On-Demand capacity of `num`. Target On-Demand capacity was `num`. For more information and recommended actions, please check the documentation page here and retry again.

Event type	Severity	Event code	Message
EMR instance fleet resize	WARNING	EC2 provisioning - Insufficient Instance Capacity	We are not able to complete the resize operation for Instance Fleet InstanceFleetID in EMR cluster ClusterId (ClusterName) as Amazon EC2 has insufficient Spot capacity for Instance types [Instancetype1, Instancetype2] and insufficient On-Demand capacity for Instance types [Instancetype3, Instancetype4] in Availability Zone [AvailabilityZone1]. So far, the instance fleet has provisioned On-Demand capacity of num and target On-Demand capacity was num. Provisioned Spot capacity is num and target Spot capacity was num. Check here documentation for more information on

Event type	Severity	Event code	Message
			how to respond to this event.
EMR instance fleet resize	`WARNING`	Spot Provisioning Timeout - Continuing Resize	We're still provisioning Spot capacity for the Instance Fleet resize operation that initiated at `time` for instance fleet ID `InstanceFleetID` in Amazon EMR cluster `ClusterId (ClusterName)` for `[Instance type1, Instancetype2]` in AZ `AvailabilityZone`. For the previous resize operation that initiated at `time`, the timeout period expired, so Amazon EMR stopped provisioning Spot capacity after adding `num` of the requested `num` instances to your instance fleet. For more information, please check the documentation page here.

Event type	Severity	Event code	Message
EMR instance fleet resize	`WARNING`	On-Demand Provisioning Timeout - Continuing Resize	We're still provisioning On-Demand capacity for the Instance Fleet resize operation that initiated at `time` for instance fleet ID `InstanceFleetID` in Amazon EMR cluster `ClusterId` `(ClusterName)` for `[Instance type1, Instancetype2]` in AZ `AvailabilityZone` . For the previous resize operation that initiated at `time`, the timeout period expired, so Amazon EMR stopped provisioning On-Demand capacity after adding `num` of the requested `num` instances to your instance fleet. For more information, please check the documentation page here.

Event type	Severity	Event code	Message
EMR instance fleet resize	`WARNING`	EC2 Provisioning - Insufficient Free Address in Subnet	We can't complete the resize operation for instance fleet `InstanceFleetID` in Amazon EMR cluster `ClusterId` `(ClusterName)` because the specified subnet [Subnet1, Subnet2] doesn't contain enough free private IP addresses to fulfill your request. Use the `DescribeSubnets` operation to view how many IP addresses are available (unused) in your subnet. For information on how to respond to this event, see Error codes for the Amazon EC2 API.

Event type	Severity	Event code	Message
EMR instance fleet resize	`WARNING`	EC2 Provisioning - vCPU Limit Exceeded	The resize of instance fleet `InstanceFleetID` in the Amazon EMR cluster `ClusterName` is delayed because you've reached the limit on the number of vCPUs (virtual processing units) assigned to the running instances in your `account (accountId)`. For more information, see Error codes for the Amazon EC2 API.
EMR instance fleet resize	`WARNING`	EC2 Provisioning - Spot Instance Count Limit Exceeded	The provision of instance fleet `InstanceFleetID` in the Amazon EMR cluster `ClusterID (ClusterName)` is delayed because you've reached the limit on the number of Spot Instances that you can launch in your `account (accountId)`. For more information, see Error codes for the Amazon EC2 API.

Event type	Severity	Event code	Message
EMR instance fleet resize	`WARNING`	EC2 Provisioning - Instance Limit Exceeded	The provision of instance fleet `InstanceFleetID` in the Amazon EMR cluster `ClusterID` `(ClusterName)` is delayed because you've reached the limit on the number of on-demand instances you can run in your `account` `(accountId)` . For more information on Error codes for the Amazon EC2 API.

> **Note**
>
> The provisioning timeout events are emitted when Amazon EMR stops provisioning Spot or On-demand capacity for the fleet after the timeout expires. For information on how to respond to these events, see Responding to Amazon EMR cluster instance fleet resize timeout events .

Instance group events

Event type	Severity	Event code	Message
From `RESIZING` to `Running`	`INFO`	*none*	The resizing operation for instance group `InstanceGroupID` in Amazon EMR

Event type	Severity	Event code	Message
			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`	*none*	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`	*none*	Instance group `InstanceGroupID` in Amazon EMR cluster `ClusterId` `(ClusterName)` was arrested at `Time` for the following reason: `ReasonDesc` .

Event type	Severity	Event code	Message
`RESIZING`	`WARNING`	*none*	The resizing operation for instance group `InstanceGroupID` in Amazon EMR cluster `ClusterId` `(Cluster Name)` is stuck for the following reason: `ReasonDesc` .

Event type	Severity	Event code	Message
EMR instance group resize	`WARNING`	EC2 provisioning - Insufficient Instance Capacity	We are not able to complete the resize operation that started at `time` for Instance Group `InstanceGroupID` in EMR cluster `ClusterId (ClusterName)` as Amazon EC2 has insufficient `Spot/On Demand` capacity for Instance type `[Instance type]` in Availability Zone `[AvailabilityZone1]` . So far, the instance group has a running instance count of `num` and requested instance count was `num`. Check here documentation for more information on how to respond to this event.

Event type	Severity	Event code	Message
EMR instance group resize	`WARNING`	EC2 Provisioning - Insufficient Free Address in Subnet	We can't complete the resize operation for instance group `InstanceGroupID` in Amazon EMR cluster `ClusterId` `(ClusterName)` because the specified subnet [Subnet1, Subnet2] doesn't contain enough free private IP addresses to fulfill your request. Use the `DescribeSubnets` operation to view how many IP addresses are available (unused) in your subnet. For information on how to respond to this event, see Error codes for the Amazon EC2 API.

Event type	Severity	Event code	Message
EMR instance group resize	`WARNING`	EC2 Provisioning - vCPU Limit Exceeded	The resize of instance group `InstanceGroupID` in the Amazon EMR cluster `ClusterName` is delayed because you've reached the limit on the number of vCPUs (virtual processing units) assigned to the running instances in your `account (accountId)`. For more information, see Error codes for the Amazon EC2 API.
EMR instance group resize	`WARNING`	EC2 Provisioning - Spot Instance Count Limit Exceeded	The provision of instance group `InstanceGroupID` in the Amazon EMR cluster `ClusterID (ClusterName)` is delayed because you've reached the limit on the number of Spot Instances that you can launch in your `account (accountId)`. For more information, see Error codes for the Amazon EC2 API.

Event type	Severity	Event code	Message
EMR instance group resize	`WARNING`	EC2 Provisioning - Instance Limit Exceeded	The provision of instance group `InstanceGroupID` in the Amazon EMR cluster `ClusterID` `(ClusterName)` is delayed because you've reached the limit on the number of on-demand instances you can run in your `account` `(accountId)` . For more information on Error codes for the Amazon EC2 API.
From `RUNNING` to `RESIZING`	`INFO`	*none*	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 in	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

State or state change	Severity	Message
		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 (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`.

State or state change	Severity	Message
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

State or state change	Severity	Message
		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 (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.
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.

State or state change	Severity	Message
FAILED	ERROR	Step StepID (StepName) in Amazon EMR cluster ClusterId (ClusterName) failed at Time.

Unhealthy node replacement events

Event type	Severity	Event code	Message	
Amazon EMR unhealthy node replacement	INFO	Unhealthy core node detected	Amazon EMR has identified that core instance [instanceID (InstanceName)] in InstanceGroup/Fleet in the Amazon EMR cluster clusterID (ClusterName) is UNHEALTHY. Amazon EMR will attempt to recover or gracefully replace the UNHEALTHY instance.	

Event type	Severity	Event code	Message	
Amazon EMR unhealthy node replacement	`INFO`	Core node unhealthy - replacement disabled	Amazon EMR has identified that core instance `[instanceID (Instance Name)]` in `InstanceGroup/Fleet` in the Amazon EMR cluster `(clusterID) (ClusterName)` is `UNHEALTHY`. Turn on graceful unhealthy core node replacement in your cluster to let Amazon EMR gracefully replace the `UNHEALTHY` instances in the event that they can't be recovered.	

Event type	Severity	Event code	Message
Amazon EMR unhealthy node replacement	WARN	Unhealthy core node not replaced	Amazon EMR can't replace your UNHEALTHY core instance [instanceID (Instance Name)] in InstanceGroup/Fleet in the Amazon EMR cluster clusterID (ClusterName) because of *reason*. **Note** The reason of why Amazon EMR can't replace your core node differs depending on your scenario. For example, one reason

Event type	Severity	Event code	Message
			of why Amazon EMR can't delete a node is because a cluster wouldn't have any remaining core nodes.
Amazon EMR unhealthy node replacement	`INFO`	Unhealthy core node recovered	Amazon EMR has recovered your `UNHEALTHY` core instances `[instanceID (Instance Name)]` in `InstanceGroup/Fleet` in the Amazon EMR cluster `clusterID (ClusterName)`

For more information about unhealthy node replacement, see Replacing unhealthy nodes.

Viewing events with 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.

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 with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Events**.

To view events for a particular cluster with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose a cluster.
3. To view all of your events, select the **Events** tab on the cluster details page.

Responding to CloudWatch events from Amazon EMR

This section describes various ways that you can respond to actionable events that Amazon EMR emits as CloudWatch event messages. Ways you can respond to events include creating rules, setting alarms, and other responses. The sections that follow include links to procedures and recommneded responses to common evens.

Topics

- Creating rules for Amazon EMR events with CloudWatch
- Setting alarms on CloudWatch metrics from Amazon EMR
- Responding to Amazon EMR cluster insufficient instance capacity events
- Responding to Amazon EMR cluster instance fleet resize timeout events

Creating rules for Amazon EMR events with 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.

Setting alarms on CloudWatch metrics from Amazon EMR

Amazon EMR pushes metrics to Amazon CloudWatch. In response, 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*.

Responding to Amazon EMR cluster insufficient instance capacity events

Overview

Amazon EMR clusters return the event code `EC2 provisioning - Insufficient Instance Capacity` when the selected Availability Zone doesn't have enough capacity to fulfill your cluster start or resize request. The event emits periodically with both instance groups and instance fleets if Amazon EMR repeatedly encounters insufficient capacity exceptions and can't fulfill your provisioning request for a cluster start or cluster resize operation.

This page describes how you can best respond to this event type when it occurs for your EMR cluster.

Recommended response to an insufficient capacity event

We recommend that you respond to an insufficient-capacity event in one of the following ways:

- Wait for capacity to recover. Capacity shifts frequently, so an insufficient capacity exception can recover on its own. Your clusters will start or finish resizing as soon as Amazon EC2 capacity becomes available.
- Alternatively, you can terminate your cluster, modify your instance type configurations, and create a new cluster with the updated cluster configuration request. For more information, see Availability Zone flexibility for an Amazon EMR cluster.

You can also set up rules or automated responses to an insufficient capacity event, as described in the next section.

Automated recovery from an insufficient capacity event

You can build automation in response to Amazon EMR events such as the ones with event code `EC2 provisioning - Insufficient Instance Capacity`. For example, the following AWS Lambda function terminates an EMR cluster with an instance group that uses On-Demand instances, and then creates a new EMR cluster with an instance group that contains different instance types than the original request.

The following conditions trigger the automated process to occur:

- The insufficient capacity event has been emitting for primary or core nodes for more than 20 minutes.
- The cluster is not in a **READY** or **WAITING** state. For more information about EMR cluster states, see Understanding the cluster lifecycle.

> **Note**
>
> When you build an automated process for an insufficient capacity exception, you should consider that the insufficient capacity event is recoverable. Capacity often shifts and your clusters will resume the resize or start operation as soon as Amazon EC2 capacity becomes available.

Example function to respond to insufficient capacity event

```
// Lambda code with Python 3.10 and handler is lambda_function.lambda_handler
// Note: related IAM role requires permission to use Amazon EMR

import json
import boto3
import datetime
from datetime import timezone

INSUFFICIENT_CAPACITY_EXCEPTION_DETAIL_TYPE = "EMR Instance Group Provisioning"
INSUFFICIENT_CAPACITY_EXCEPTION_EVENT_CODE = (
    "EC2 provisioning - Insufficient Instance Capacity"
)
```

```
ALLOWED_INSTANCE_TYPES_TO_USE = [
    "m5.xlarge",
    "c5.xlarge",
    "m5.4xlarge",
    "m5.2xlarge",
    "t3.xlarge",
]
CLUSTER_START_ACCEPTABLE_STATES = ["WAITING", "RUNNING"]
CLUSTER_START_SLA = 20

CLIENT = boto3.client("emr", region_name="us-east-1")

# checks if the incoming event is 'EMR Instance Fleet Provisioning' with eventCode 'EC2
 provisioning - Insufficient Instance Capacity'
def is_insufficient_capacity_event(event):
    if not event["detail"]:
        return False
    else:
        return (
            event["detail-type"] == INSUFFICIENT_CAPACITY_EXCEPTION_DETAIL_TYPE
            and event["detail"]["eventCode"]
            == INSUFFICIENT_CAPACITY_EXCEPTION_EVENT_CODE
        )


# checks if the cluster is eligible for termination
def is_cluster_eligible_for_termination(event, describeClusterResponse):
    # instanceGroupType could be CORE, MASTER OR TASK
    instanceGroupType = event["detail"]["instanceGroupType"]
    clusterCreationTime = describeClusterResponse["Cluster"]["Status"]["Timeline"][
        "CreationDateTime"
    ]
    clusterState = describeClusterResponse["Cluster"]["Status"]["State"]

    now = datetime.datetime.now()
    now = now.replace(tzinfo=timezone.utc)
    isClusterStartSlaBreached = clusterCreationTime < now - datetime.timedelta(
        minutes=CLUSTER_START_SLA
    )

    # Check if instance group receiving Insufficient capacity exception is CORE or
 PRIMARY (MASTER),
    # and it's been more than 20 minutes since cluster was created but the cluster
 state and the cluster state is not updated to RUNNING or WAITING
```

```
    if (
        (instanceGroupType == "CORE" or instanceGroupType == "MASTER")
        and isClusterStartSlaBreached
        and clusterState not in CLUSTER_START_ACCEPTABLE_STATES
    ):
        return True
    else:
        return False


# Choose item from the list except the exempt value
def choice_excluding(exempt):
    for i in ALLOWED_INSTANCE_TYPES_TO_USE:
        if i != exempt:
            return i


# Create a new cluster by choosing different InstanceType.
def create_cluster(event):
    # instanceGroupType cloud be CORE, MASTER OR TASK
    instanceGroupType = event["detail"]["instanceGroupType"]

    # Following two lines assumes that the customer that created the cluster already
 knows which instance types they use in original request
    instanceTypesFromOriginalRequestMaster = "m5.xlarge"
    instanceTypesFromOriginalRequestCore = "m5.xlarge"

    # Select new instance types to include in the new createCluster request
    instanceTypeForMaster = (
        instanceTypesFromOriginalRequestMaster
        if instanceGroupType != "MASTER"
        else choice_excluding(instanceTypesFromOriginalRequestMaster)
    )
    instanceTypeForCore = (
        instanceTypesFromOriginalRequestCore
        if instanceGroupType != "CORE"
        else choice_excluding(instanceTypesFromOriginalRequestCore)
    )

    print("Starting to create cluster...")
    instances = {
        "InstanceGroups": [
            {
                "InstanceRole": "MASTER",
```

```
                "InstanceCount": 1,
                "InstanceType": instanceTypeForMaster,
                "Market": "ON_DEMAND",
                "Name": "Master",
            },
            {
                "InstanceRole": "CORE",
                "InstanceCount": 1,
                "InstanceType": instanceTypeForCore,
                "Market": "ON_DEMAND",
                "Name": "Core",
            },
        ]
    }
    response = CLIENT.run_job_flow(
        Name="Test Cluster",
        Instances=instances,
        VisibleToAllUsers=True,
        JobFlowRole="EMR_EC2_DefaultRole",
        ServiceRole="EMR_DefaultRole",
        ReleaseLabel="emr-6.10.0",
    )

    return response["JobFlowId"]


# Terminated the cluster using clusterId received in an event
def terminate_cluster(event):
    print("Trying to terminate cluster, clusterId: " + event["detail"]["clusterId"])
    response = CLIENT.terminate_job_flows(JobFlowIds=[event["detail"]["clusterId"]])
    print(f"Terminate cluster response: {response}")


def describe_cluster(event):
    response = CLIENT.describe_cluster(ClusterId=event["detail"]["clusterId"])
    return response


def lambda_handler(event, context):
    if is_insufficient_capacity_event(event):
        print(
            "Received insufficient capacity event for instanceGroup, clusterId: "
            + event["detail"]["clusterId"]
        )
```

```
        describeClusterResponse = describe_cluster(event)

        shouldTerminateCluster = is_cluster_eligible_for_termination(
            event, describeClusterResponse
        )
        if shouldTerminateCluster:
            terminate_cluster(event)

            clusterId = create_cluster(event)
            print("Created a new cluster, clusterId: " + clusterId)
        else:
            print(
                "Cluster is not eligible for termination, clusterId: "
                + event["detail"]["clusterId"]
            )

    else:
        print("Received event is not insufficient capacity event, skipping")
```

Responding to Amazon EMR cluster instance fleet resize timeout events

Overview

Amazon EMR clusters emit events while executing the resize operation for instance fleet clusters. The provisioning timeout events are emitted when Amazon EMR stops provisioning Spot or On-demand capacity for the fleet after the timeout expires. The timeout duration can be configured by the user as part of the resize specifications for the instance fleets. In scenarios of consecutive resizes for the same instance fleet, Amazon EMR emits the `Spot provisioning timeout - continuing resize` or `On-Demand provisioning timeout - continuing resize` events when timeout for the current resize operation expires. It then starts provisioning capacity for the fleet's next resize operation.

Responding to instance fleet resize timeout events

We recommend that you respond to a provisioning timeout event in one of the following ways:

- Revisit the resize specifications and retry the resize operation. As capacity shifts frequently, your clusters will successfully resize as soon as Amazon EC2 capacity becomes available. We recommend customers to configure lower values for the timeout duration for the jobs that require stricter SLAs.
- Alternatively, you can either:

- Launch a new cluster with diversified instance types based on the best practices for instance and Availability Zone flexibility or
- Launch a cluster with On-demand capacity
- For the provisioning timeout - continuing resize event, you can additionally wait for resize operations to be processed. Amazon EMR will continue to sequentially process the resize operations triggered for the fleet, respecting the configured resize specifications.

You can also set up rules or automated responses to this event as described in the next section.

Automated recovery from a provisioning timeout event

You can build automation in response to Amazon EMR events with the `Spot Provisioning timeout` event code. For example, the following AWS Lambda function shuts down an EMR cluster with an instance fleet that uses Spot instances for Task nodes, and then creates a new EMR cluster with an instance fleet that contains more diversified instance types than the original request. In this example, the `Spot Provisioning timeout` event emitted for task nodes will trigger the execution of the Lambda function.

Example Example function to respond to `Spot Provisioning timeout` event

```
// Lambda code with Python 3.10 and handler is lambda_function.lambda_handler
// Note: related IAM role requires permission to use Amazon EMR

import json
import boto3
import datetime
from datetime import timezone

SPOT_PROVISIONING_TIMEOUT_EXCEPTION_DETAIL_TYPE = "EMR Instance Fleet Resize"
SPOT_PROVISIONING_TIMEOUT_EXCEPTION_EVENT_CODE = (
    "Spot Provisioning timeout"
)

CLIENT = boto3.client("emr", region_name="us-east-1")

# checks if the incoming event is 'EMR Instance Fleet Resize' with eventCode 'Spot
 provisioning timeout'
def is_spot_provisioning_timeout_event(event):
    if not event["detail"]:
        return False
```

```
    else:
        return (
            event["detail-type"] == SPOT_PROVISIONING_TIMEOUT_EXCEPTION_DETAIL_TYPE
            and event["detail"]["eventCode"]
            == SPOT_PROVISIONING_TIMEOUT_EXCEPTION_EVENT_CODE
        )


# checks if the cluster is eligible for termination
def is_cluster_eligible_for_termination(event, describeClusterResponse):
    # instanceFleetType could be CORE, MASTER OR TASK
    instanceFleetType = event["detail"]["instanceFleetType"]

    # Check if instance fleet receiving Spot provisioning timeout event is TASK
    if (instanceFleetType == "TASK"):
        return True
    else:
        return False


# create a new cluster by choosing different InstanceType.
def create_cluster(event):
    # instanceFleetType cloud be CORE, MASTER OR TASK
    instanceFleetType = event["detail"]["instanceFleetType"]

    # the following two lines assumes that the customer that created the cluster
 already knows which instance types they use in original request
    instanceTypesFromOriginalRequestMaster = "m5.xlarge"
    instanceTypesFromOriginalRequestCore = "m5.xlarge"

    # select new instance types to include in the new createCluster request
    instanceTypesForTask = [
        "m5.xlarge",
        "m5.2xlarge",
        "m5.4xlarge",
        "m5.8xlarge",
        "m5.12xlarge"
    ]

    print("Starting to create cluster...")
    instances = {
        "InstanceFleets": [
            {
                "InstanceFleetType":"MASTER",
```

```
                "TargetOnDemandCapacity":1,
                "TargetSpotCapacity":0,
                "InstanceTypeConfigs":[
                    {
                        'InstanceType': instanceTypesFromOriginalRequestMaster,
                        "WeightedCapacity":1,
                    }
                ]
            },
            {
                "InstanceFleetType":"CORE",
                "TargetOnDemandCapacity":1,
                "TargetSpotCapacity":0,
                "InstanceTypeConfigs":[
                    {
                        'InstanceType': instanceTypesFromOriginalRequestCore,
                        "WeightedCapacity":1,
                    }
                ]
            },
            {
                "InstanceFleetType":"TASK",
                "TargetOnDemandCapacity":0,
                "TargetSpotCapacity":100,
                "LaunchSpecifications":{},
                "InstanceTypeConfigs":[
                    {
                        'InstanceType': instanceTypesForTask[0],
                        "WeightedCapacity":1,
                    },
                    {
                        'InstanceType': instanceTypesForTask[1],
                        "WeightedCapacity":2,
                    },
                    {
                        'InstanceType': instanceTypesForTask[2],
                        "WeightedCapacity":4,
                    },
                    {
                        'InstanceType': instanceTypesForTask[3],
                        "WeightedCapacity":8,
                    },
                    {
                        'InstanceType': instanceTypesForTask[4],
```

```
                        "WeightedCapacity":12,
                    }
                ],
                "ResizeSpecifications": {
                    "SpotResizeSpecification": {
                        "TimeoutDurationMinutes": 30
                    }
                }
            }
        ]
    }
    response = CLIENT.run_job_flow(
        Name="Test Cluster",
        Instances=instances,
        VisibleToAllUsers=True,
        JobFlowRole="EMR_EC2_DefaultRole",
        ServiceRole="EMR_DefaultRole",
        ReleaseLabel="emr-6.10.0",
    )

    return response["JobFlowId"]


# terminated the cluster using clusterId received in an event
def terminate_cluster(event):
    print("Trying to terminate cluster, clusterId: " + event["detail"]["clusterId"])
    response = CLIENT.terminate_job_flows(JobFlowIds=[event["detail"]["clusterId"]])
    print(f"Terminate cluster response: {response}")


def describe_cluster(event):
    response = CLIENT.describe_cluster(ClusterId=event["detail"]["clusterId"])
    return response


def lambda_handler(event, context):
    if is_spot_provisioning_timeout_event(event):
        print(
            "Received spot provisioning timeout event for instanceFleet, clusterId: "
            + event["detail"]["clusterId"]
        )

        describeClusterResponse = describe_cluster(event)
```

```
        shouldTerminateCluster = is_cluster_eligible_for_termination(
            event, describeClusterResponse
        )
        if shouldTerminateCluster:
            terminate_cluster(event)

            clusterId = create_cluster(event)
            print("Created a new cluster, clusterId: " + clusterId)
        else:
            print(
                "Cluster is not eligible for termination, clusterId: "
                + event["detail"]["clusterId"]
            )

    else:
        print("Received event is not spot provisioning timeout event, skipping")
```

View cluster application metrics using Ganglia with Amazon EMR

Ganglia is available with Amazon EMR releases between 4.2 and 6.15. 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 AWS EMR API calls using AWS CloudTrail

AWS EMR is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service. CloudTrail captures all API calls for AWS EMR as events. The calls captured include calls from the AWS EMR console and code calls to the AWS EMR API operations. Using the information collected by CloudTrail, you can determine the request that was made to AWS EMR, the IP address from which the request was made, when it was made, and additional details.

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 user or user credentials.
- Whether the request was made on behalf of an IAM Identity Center user.

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

CloudTrail is active in your AWS account when you create the account and you automatically have access to the CloudTrail **Event history**. The CloudTrail **Event history** provides a viewable, searchable, downloadable, and immutable record of the past 90 days of recorded management events in an AWS Region. For more information, see Working with CloudTrail Event history in the *AWS CloudTrail User Guide*. There are no CloudTrail charges for viewing the **Event history**.

For an ongoing record of events in your AWS account past 90 days, create a trail or a CloudTrail Lake event data store.

CloudTrail trails

A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. All trails created using the AWS Management Console are multi-Region. You can create a single-Region or a multi-Region trail by using the AWS CLI. Creating a multi-Region trail is recommended because you capture activity in all AWS Regions in your account. If you create a single-Region trail, you can view only the events logged in the trail's AWS Region. For more information about trails, see Creating a trail for your AWS account and Creating a trail for an organization in the *AWS CloudTrail User Guide*.

You can deliver one copy of your ongoing management events to your Amazon S3 bucket at no charge from CloudTrail by creating a trail, however, there are Amazon S3 storage charges. For more information about CloudTrail pricing, see AWS CloudTrail Pricing. For information about Amazon S3 pricing, see Amazon S3 Pricing.

CloudTrail Lake event data stores

CloudTrail Lake lets you run SQL-based queries on your events. CloudTrail Lake converts existing events in row-based JSON format to Apache ORC format. ORC is a columnar storage format that is optimized for fast retrieval of data. Events are aggregated into *event data stores*, which are immutable collections of events based on criteria that you select by applying advanced event selectors. The selectors that you apply to an event data store control which events persist and are available for you to query. For more information about CloudTrail Lake, see Working with AWS CloudTrail Lake in the *AWS CloudTrail User Guide*.

CloudTrail Lake event data stores and queries incur costs. When you create an event data store, you choose the pricing option you want to use for the event data store. The pricing

option determines the cost for ingesting and storing events, and the default and maximum retention period for the event data store. For more information about CloudTrail pricing, see AWS CloudTrail Pricing.

AWS EMR data events in CloudTrail

Data events provide information about the resource operations performed on or in a resource (for example, reading or writing to an Amazon S3 object). These are also known as data plane operations. Data events are often high-volume activities. By default, CloudTrail doesn't log data events. The CloudTrail **Event history** doesn't record data events.

Additional charges apply for data events. For more information about CloudTrail pricing, see AWS CloudTrail Pricing.

You can log data events for the AWS EMR resource types by using the CloudTrail console, AWS CLI, or CloudTrail API operations. For more information about how to log data events, see Logging data events with the AWS Management Console and Logging data events with the AWS Command Line Interface in the *AWS CloudTrail User Guide*.

The following table lists the AWS EMR resource types for which you can log data events. The **Data event type (console)** column shows the value to choose from the **Data event type** list on the CloudTrail console. The **resources.type value** column shows the `resources.type` value, which you would specify when configuring advanced event selectors using the AWS CLI or CloudTrail APIs. The **Data APIs logged to CloudTrail** column shows the API calls logged to CloudTrail for the resource type.

For more information about these API operations, see Amazon EMR WAL (EMRWAL) CLI reference. Amazon EMR logs some Data API operations to CloudTrail that are HBase system operations that you never call directly. These operations aren't in the EMRWAL CLI reference.

Data event type (console)	resources.type value	Data APIs logged to CloudTrail
Amazon EMR write-ahead log workspace	`AWS::EMRWAL::Workspace`	• GetCurrentWALTime • ListTagsForResource • ListWALs • ListWorkspaces

Data event type (console)	resources.type value	Data APIs logged to CloudTrail
		• TrimWAL • CompleteWALFlush

You can configure advanced event selectors to filter on the `eventName`, `readOnly`, and `resources.ARN` fields to log only those events that are important to you. For more information about these fields, see AdvancedFieldSelector in the *AWS CloudTrail API Reference*.

AWS EMR management events in CloudTrail

Management events provide information about management operations that are performed on resources in your AWS account. These are also known as control plane operations. By default, CloudTrail logs management events.

AWS EMR logs all AWS EMR control plane operations as management events. For a list of the AWS EMR control plane operations that AWS EMR logs to CloudTrail, see the AWS EMR API Reference.

AWS EMR event examples

An event represents a single request from any source and includes information about the requested API operation, the date and time of the operation, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so events 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
  ]
}
```

For information about CloudTrail record contents, see CloudTrail record contents in the *AWS CloudTrail User Guide*.

EMR Observability Best Practices

EMR Observability encompasses a comprehensive monitoring and management approach for AWS EMR clusters. The foundation rests on Amazon CloudWatch as the primary monitoring service, complemented by EMR Studio, and third-party tools like Prometheus and Grafana for enhanced visibility. In this document, we explore specific aspects of cluster observability:

1. *Spark observability* (GitHub) – With regards to the Spark user interface, you have three options in Amazon EMR.
2. *Spark troubleshooting* (GitHub) – Resolutions for errors.
3. *EMR Cluster monitoring* (GitHub) – Monitoring cluster performance.
4. *Troubleshooting EMR* (GitHub) – Identify, diagnose, and resolve common EMR cluster problems.
5. *Cost optimization* (GitHub) – This section outlines the best practices for running cost-effective workloads.

Performance Optimization Tool for Apache Spark Applications

1. AWS EMR Advisor tool analyzes Spark event logs to provide tailored recommendations for optimizing EMR cluster configurations, enhancing performance, and reducing costs. By leveraging historical data, it suggests ideal executor sizes and infrastructure settings, enabling more efficient resource utilization and improved overall cluster performance.
2. Amazon CodeGuru Profiler tool helps developers identify performance bottlenecks and inefficiencies in their Spark applications by collecting and analyzing runtime data. The tool integrates seamlessly with existing Spark applications, requiring minimal setup, and provides detailed insights through the AWS Console about CPU usage, memory patterns, and performance hotspots.

Use Amazon EMR cluster scaling to adjust for changing workloads

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 Amazon EMR managed scaling or create a custom automatic scaling policy. The following table describes the differences between the two options.

	Amazon EMR managed scaling	**Custom automatic scaling**
Scaling policies and rules	No policy required. Amazon 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 Amazon EMR releases	Amazon EMR version 5.30.0 and higher (except Amazon EMR version 6.0.0)	Amazon EMR version 4.0.0 and higher
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 Amazon 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. Amazon EMR managed scaling does not support applications that	You can choose which applications are supported when defining the automatic scaling rules.

	Amazon EMR managed scaling	Custom automatic scaling
	are not based on YARN, such as Presto or HBase.	

Considerations

- An Amazon EMR cluster always comprises one or three primary nodes. Once you initially configure the cluster, you can only scale core and task nodes. You can't scale the number of primary nodes for the cluster.
- For instance groups, reconfiguration operations and resize operations occur consecutively and not concurrently. If you initiate a reconfiguration while an instance group is resizing, the reconfiguration starts once the instance group completes the resize in progress. Conversely, if you initiate a resize operation while an instance group its reconfiguration.

Using managed scaling in Amazon EMR

> **Important**
>
> We strongly recommend that you use the latest Amazon EMR release (Amazon EMR 7.10.0) for managed scaling. In some early releases, you might experience intermittent application failures or delays in scaling. Amazon EMR resolved this issue with 5.x releases 5.30.2, 5.31.1, 5.32.1, 5.33.1 and higher, and with 6.x releases 6.1.1, 6.2.1, 6.3.1 and higher. For more information Region and release availability, see Managed scaling availability.

Overview

With Amazon EMR versions 5.30.0 and higher (except for Amazon EMR 6.0.0), you can enable Amazon EMR managed scaling. Managed scaling lets you automatically increase or decrease the number of instances or units in your cluster based on workload. Amazon 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.

Managed scaling availability

- In the following AWS Regions, Amazon EMR managed scaling is available with Amazon EMR 6.14.0 and higher:
  - Europe (Spain) (eu-south-2)
- In the following AWS Regions, Amazon EMR managed scaling is available with Amazon EMR 5.30.0 and 6.1.0 and higher:
  - US East (N. Virginia) (us-east-1)
  - US East (Ohio) (us-east-2)
  - US West (Oregon) (us-west-2)
  - US West (N. California) (us-west-1)
  - Africa (Cape Town) (af-south-1)
  - Asia Pacific (Hong Kong) (ap-east-1)
  - Asia Pacific (Mumbai) (ap-south-1)
  - Asia Pacific (Hyderabad) (ap-south-2)
  - Asia Pacific (Seoul) (ap-northeast-2)
  - Asia Pacific (Singapore) (ap-southeast-1)
  - Asia Pacific (Sydney) (ap-southeast-2)
  - Asia Pacific (Jakarta) (ap-southeast-3)
  - Asia Pacific (Tokyo) (ap-northeast-1)
  - Asia Pacific (Osaka) (ap-northeast-3)
  - Canada (Central) (ca-central-1)
  - South America (São Paulo) (sa-east-1)
  - Europe (Frankfurt) (eu-central-1)
  - Europe (Zurich) (eu-central-2)
  - Europe (Ireland) (eu-west-1)
  - Europe (London) (eu-west-2)
  - Europe (Milan) (eu-south-1)
  - Europe (Paris) (eu-west-3)
  - Europe (Stockholm) (eu-north-1)

  - Israel (Tel Aviv) (il-central-1)

- Middle East (UAE) (me-central-1)
- China (Beijing) (cn-north-1)
- China (Ningxia) (cn-northwest-1)
- AWS GovCloud (US-East) (us-gov-east-1)
- AWS GovCloud (US-West) (us-gov-west-1)

- Amazon EMR managed scaling only works with YARN applications, such as Spark, Hadoop, Hive, and Flink. It doesn't support applications that are not based on YARN, such as Presto and HBase.

Managed scaling parameters

You must configure the following parameters for managed scaling. The limit only applies to the core and task nodes. You cannot scale the primary node 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 Amazon 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.
- **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 Amazon 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.

For more information about managed scaling parameters, see `ComputeLimits`.

Considerations for Amazon EMR managed scaling

- Managed scaling is supported in limited AWS Regions and Amazon EMR releases. For more information, see Managed scaling availability.
- You must configure the required parameters for Amazon EMR managed scaling. For more information, see Managed scaling parameters.
- 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.
- 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.

  The following example YARN Resource Manager Log line shows nodes 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: []
```

See 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, and configuring Spark node-decommissioning behavior.

- For Spark workloads, disabling Spark Dynamic Resource Allocator (DRA) by changing the Spark property **spark.dynamicAllocation.enabled** to `FALSE` can cause Managed Scaling issues, where clusters can be scaled up more than required for your workloads (up to the maximum compute). When using Managed Scaling for these workloads, we recommend that you keep Spark DRA enabled, which is the default state of this property.
- Over-utilization of EBS volumes can cause Managed Scaling issues. We recommend that you maintain EBS volume below 90% utilization. For more information, see Instance storage options and behavior in Amazon EMR.
- 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.
- When you set the maximum core node and the On-Demand limit for Amazon 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.

- With Amazon EMR 5.30.0 and higher, 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 primary instance (private subnets).
- You can use AWS CloudFormation to configure Amazon EMR managed scaling. For more information, see AWS::EMR::Cluster in the *AWS CloudFormation User Guide*.
- If you're using Spot nodes, consider using node labels to prevent Amazon EMR from removing application processes when Amazon EMR removes Spot nodes. For more information about node labels, see Task nodes.
- Node labeling is not supported by default in Amazon EMR releases 6.15 or lower. For more information, see Understand node types: primary, core, and task nodes.
- If you're using Amazon EMR releases 6.15 or lower, you can only assign node labels by node type, such as core and task nodes. However, if you're using Amazon EMR release 7.0 or higher, you can configure node labels by node type and market type, such as On-Demand and Spot.
- If application process demand increases and executor demand decreases when you restricted the application process to core nodes, you can add back core nodes and remove task nodes in the same resize operation. For more information, see Understanding node allocation strategy and scenarios.
- Amazon EMR doesn't label task nodes, so you can't set the YARN properties to restrict application processes only for task nodes. However, if you want to use market types as node labels, you can use the `ON_DEMAND` or `SPOT` labels for application process placement. We don't recommend using Spot nodes for application primary processes.
- When using node labels, the total running units in the cluster can temporarily exceed the max compute set in your managed scaling policy while Amazon EMR decommissions some of your instances. Total requested units will always stay at or below your policy’s max compute.
- Managed scaling only supports the node labels `ON_DEMAND` and `SPOT` or `CORE` and `TASK`. Custom node labels aren't supported.
- Amazon EMR creates node labels when creating the cluster and provisioning resources. Amazon EMR doesn't support adding node labels when you reconfigure the cluster. You also can't modify the node labels when configuring managed scaling after launching the cluster.
- Managed scaling scales core and task nodes independently based on application process and executor demand. To prevent HDFS data loss issues during core scale down, follow standard

practice for core nodes. To learn more about best practices about core nodes and HDFS replication, see Considerations and best practices.

- You can't place both the application process and executors on only the `core` or the `ON_DEMAND` node. If you want to add both the application process and executors on one of the nodes, don't use the `yarn.node-labels.am.default-node-label-expression` configuration.

  For example, to place both the application process and executors in `ON_DEMAND` nodes, set max compute to the same as the maximum in the `ON_DEMAND` node. Also remove the `yarn.node-labels.am.default-node-label-expression` configuration.

  To add both the application process and executors on `core` nodes, remove the `yarn.node-labels.am.default-node-label-expression` configuration.
- When you use managed scaling with node labels, set the property `yarn.scheduler.capacity.maximum-am-resource-percent: 1` if you plan to run multiple applications in parallel. Doing so ensures that your application processes fully utilize the available `CORE` or `ON_DEMAND` nodes.
- If you use managed scaling with node labels, set the property `yarn.resourcemanager.decommissioning.timeout` to a value that's longer than the longest running application on your cluster. Doing so reduces the chance that Amazon EMR managed scalling needs to reschedule your applications to recommission `CORE` or `ON_DEMAND` nodes.
- To reduce the risk of application failures due to shuffle data loss, Amazon EMR collects metrics from the cluster to determine nodes that have existing transient shuffle data from the current and previous stage. In rare cases, metrics can continue to report stale data for applications that are already completed or terminated. This can impact timely scale down of instances in your cluster. For clusters that have large amount of shuffle data, consider using EMR versions 6.13 and later.

Feature history

This table lists updates to the Amazon EMR managed scaling capability.

Release date	Capability	Amazon EMR versions
November 20, 2024	Managed scaling is available in the `il-central-1` Israel	5.30.0 and 6.1.0 and higher

Release date	Capability	Amazon EMR versions
	(Tel Aviv), `me-central-1` Middle East (UAE), and `ap-northeast-3` Asia Pacific (Osaka) regions.	
November 15, 2024	Managed scaling is available in the `eu-central-2` Europe (Zurich) Region.	5.30.0 and 6.1.0 and higher
August 20, 2024	Node labels are now available in managed scaling, so you can label your instances b ased on market type or node type to improve automatic scaling.	7.2.0 and higher
March 31, 2024	Managed scaling is available in the `ap-south-2` Asia Pacific (Hyderabad) Region.	6.14.0 and higher
February 13, 2024	Managed scaling is available in the `eu-south-2` Europe (Spain) Region.	6.14.0 and higher
October 10, 2023	Managed scaling is available in the `ap-southeast-3` Asia Pacific (Jakarta) Region.	6.14.0 and higher
July 28, 2023	Enhanced managed scaling to switch to different task instance group on scale-up when Amazon EMR experienc es a delay in scale-up with the current instance group.	5.34.0 and higher, 6.4.0 and higher

Release date	Capability	Amazon EMR versions
June 16, 2023	Enhanced managed scaling to be aware of the nodes running application master so that those nodes are not scaled down. For more information, see Understanding Amazon EMR node allocation strategy and scenarios.	5.34.0 and higher, 6.4.0 and higher
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 higher, 6.4.0 and higher

Configure managed scaling for Amazon EMR

The following sections explain how to launch an EMR cluster that uses managed scaling with the AWS Management Console, the AWS SDK for Java, or the AWS Command Line Interface.

Topics

- Use the AWS Management Console to configure managed scaling
- Use the AWS CLI to configure managed scaling
- Use AWS SDK for Java to configure managed scaling

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.

Console

To configure managed scaling when you create a cluster with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose **Create cluster**.
3. Choose an Amazon EMR release **emr-5.30.0** or later, except version **emr-6.0.0**.
4. Under **Cluster scaling and provisioning option**, choose **Use EMR-managed scaling**. Specify the **Minimum** and **Maximum** number of instances, the **Maximum core node** instances, and the **Maximum On-Demand** instances.
5. Choose any other options that apply to your cluster.
6. To launch your cluster, choose **Create cluster**.

To configure managed scaling on an existing cluster with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and select the cluster that you want to update.
3. On the **Instances** tab of the cluster details page, find the **Instance group settings** section. Select **Edit cluster scaling** to specify new values for the **Minimum** and **Maximum** number of instances and the **On-Demand** limit.

Use 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-7.10.0 \
 --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
```

Use 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 Amazon EMR client with 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 - Primary, 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 Amazon 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-7.10.0")          // Specifies the version label for
the Amazon EMR release; 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 Amazon EMR
role for EC2 instance profile, replace the default role with the custom Amazon 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() { }
}
```

Advanced Scaling for Amazon EMR

Starting with Amazon EMR on EC2 version 7.0, you can leverage Advanced Scaling to control your cluster's resource utilization. Advanced scaling introduces a utilization-performance scale for tuning your resource utilization and performance level according to your business needs. The value you set determines whether your cluster is weighted more to resource conservation or to scaling up to handle service-level-agreement (SLA) sensitive workloads, where quick completion is critical. When the scaling value is adjusted, managed scaling interprets your intent and intelligently scales to optimize resources. For more information about managed scaling, see Configure managed scaling for Amazon EMR.

Advanced scaling settings

The value your set for Advanced Scaling optimizes your cluster to your requirements. Values range from **1**-**100**. Possible values are **1**, **25**, **50**, **75** and **100**. If you set the index to values other than these, it results in a validation error.

Scaling values map to resource-utilization strategies. The following list defines several of these:

- **Utilization optimized [1]** – This setting prevents resource over provisioning. Use a low value when you want to keep costs low and to prioritize efficient resource utilization. It causes the cluster to scale up less aggressively. This works well for the use case when there are regularly occurring workload spikes and you don't want resources to ramp up too quickly.
- **Balanced [50]** – This balances resource utilization and job performance. This setting is suitable for steady workloads where most stages have a stable runtime. It's also suitable for workloads with a mix of short and long-running stages. We recommend starting with this setting if you aren't sure which to choose.
- **Performance optimized [100]** – This strategy prioritizes performance. The cluster scales up aggressively to ensure that jobs complete quickly and meet performance targets. Performance optimized is suitable for service-level-agreement (SLA) sensitive workloads where fast run time is critical.

> **Note**
>
> The intermediate values available provide a middle ground between strategies in order to fine tune your cluster's Advanced Scaling behavior.

Benefits of advanced scaling

As you have variability in your environment and requirements, such as changing data volumes, cost-target adjustments, and SLA implementations, cluster scaling can help you adjust your cluster configuration to achieve your objectives. Key benefits include:

- **Enhanced granular control** – The introduction of the utilization-performance setting allows you to easily adjust your cluster's scaling behavior according to your requirements. You can scale up to meet demand for compute resources or scale down to save resources, based on your use patterns.
- **Improved cost optimization** – You can choose a low utilization value as requirements dictate to more easily meet your cost objectives.

Getting started with optimization

Setup and configuration

Use these steps to set the performance index and optimize your scaling strategy.

1. The following command updates an existing cluster with the utilization-optimized `[1]` scaling strategy:

```
aws emr put-managed-scaling-policy --cluster-id 'cluster-id' \
 --managed-scaling-policy '{
  "ComputeLimits": {
    "UnitType": "Instances",
    "MinimumCapacityUnits": 1,
    "MaximumCapacityUnits": 2,
    "MaximumOnDemandCapacityUnits": 2,
    "MaximumCoreCapacityUnits": 2
  },
  "ScalingStrategy": "ADVANCED",
  "UtilizationPerformanceIndex": "1"
}' \
```

```
 --region "region-name"
```

The attributes `ScalingStrategy` and `UtilizationPerformanceIndex` are new and relevant to scaling optimization. You can select different scaling strategies by setting corresponding values (1, 25, 50, 75, and 100) for the `UtilizationPerformanceIndex` attribute in the managed-scaling policy.

2. To revert to the default managed-scaling strategy, run the `put-managed-scaling-policy` command without including the `ScalingStrategy` and `UtilizationPerformanceIndex` attributes. (This is optional.) This sample shows how to do this:

```
aws emr put-managed-scaling-policy \
--cluster-id 'cluster-id' \
--managed-scaling-policy '{"ComputeLimits":
{"UnitType":"Instances","MinimumCapacityUnits":1,"MaximumCapacityUnits":2,"MaximumOnDemandC
 \
--region "region-name"
```

Using monitoring metrics to track cluster utilization

Starting with EMR version 7.3.0, Amazon EMR publishes four new metrics related to memory and virtual CPU. You can use these to measure cluster utilization across scaling strategies. These metrics are available for any use case, but you can use the details provided here for monitoring Advanced Scaling.

Helpful metrics available include the following:

- **YarnContainersUsedMemoryGBSeconds** – Amount of memory consumed by applications managed by YARN.
- **YarnContainersTotalMemoryGBSeconds** – Total memory capacity allocated to YARN within the cluster.
- **YarnNodesUsedVCPUSeconds** – Total VCPU seconds for each application managed by YARN.
- **YarnNodesTotalVCPUSeconds** – Aggregated total VCPU seconds for memory consumed, including the time window when yarn is not ready.

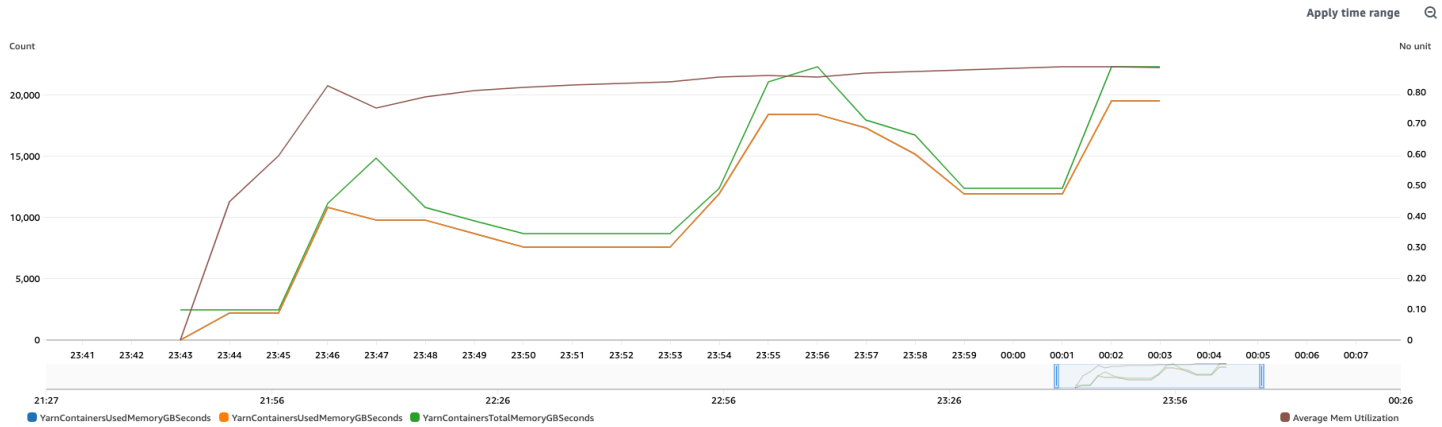
You can analyze resource metrics using Amazon CloudWatch Logs Insights. Features include a purpose-built query language that helps you extract metrics specific to resource use and scaling.

The following query, which you can run in the Amazon CloudWatch console, uses metric math to calculate the average memory utilization (e1) by dividing the running sum of consumed memory (e2) by the running sum of total memory (e3):

```
{
    "metrics": [
        [ { "expression": "e2/e3", "label": "Average Mem Utilization", "id": "e1",
 "yAxis": "right" } ],
        [ { "expression": "RUNNING_SUM(m1)", "label": "RunningTotal-
YarnContainersUsedMemoryGBSeconds", "id": "e2", "visible": false } ],
        [ { "expression": "RUNNING_SUM(m2)", "label": "RunningTotal-
YarnContainersTotalMemoryGBSeconds", "id": "e3", "visible": false } ],
        [ "AWS_EMR_ManagedResize", "YarnContainersUsedMemoryGBSeconds", "ACCOUNT_ID",
 "793684541905", "COMPONENT", "ManagerService", "JOB_FLOW_ID", "cluster-id", { "id":
 "m1", "label": "YarnContainersUsedMemoryGBSeconds" } ],
        [ ".", "YarnContainersTotalMemoryGBSeconds", ".", ".", ".", ".", ".", ".",
 { "id": "m2", "label": "YarnContainersTotalMemoryGBSeconds" } ]
    ],
    "view": "timeSeries",
    "stacked": false,
    "region": "region",
    "period": 60,
    "stat": "Sum",
    "title": "Memory Utilization"
}
```

To query logs, you can select CloudWatch in the AWS console. For more information about writing queries for CloudWatch, see Analyzing log data with CloudWatch Logs Insights in the Amazon CloudWatch Logs User Guide.

The following image shows these metrics for a sample cluster:



Considerations and limitations

- The effectiveness of scaling strategies might vary, depending on your unique workload characteristics and cluster configuration. We encourage you to experiment with the scaling setting to determine an optimal index value for your use case.
- Amazon EMR Advanced Scaling is particularly well suited for batch workloads. For SQL/data-warehousing and streaming workloads, we recommend using the default managed-scaling strategy for optimal performance.
- The performance-optimized scaling strategy enables faster job execution by maintaining high compute resources for a longer period than the default managed-scaling strategy. This mode prioritizes quickly scaling up to meet resource demands, resulting in quicker job completion. This might result in higher costs when compared with the default strategy.
- In cases where the cluster is already optimized and fully utilized, enabling Advanced Scaling might not provide additional benefits. In some situations, enabling Advanced Scaling might lead to increased costs as workloads may run longer. In these cases, we recommend using the default managed-scaling strategy to ensure optimal resource allocation and cost efficiency.
- In the context of managed scaling, the emphasis shifts towards resource utilization over execution time as the setting is adjusted from performance-optimized [**100**] to utilization-optimized [**1**]. However, it is important to note that the outcomes might vary, based on the nature of the workload and the cluster's topology. To ensure optimal results for your use case, we strongly recommend testing the scaling strategies with your workloads to determine the most suitable setting.
- The **PerformanceUtilizationIndex** accepts only the following values:
  - **1**
  - **25**
  - **50**
  - **75**
  - **100**

  Any other values submitted result in a validation error.

Understanding Amazon EMR node allocation strategy and scenarios

This section gives an overview of node allocation strategy and common scaling scenarios that you can use with Amazon EMR managed scaling.

Node allocation strategy

Amazon EMR managed scaling allocates core and task nodes based on the following scale-up and scale-down strategies:

Scale-up strategy

- For Amazon EMR releases 7.2 and higher, managed scaling first adds nodes based on node labels and the application process restriction YARN property.
- For Amazon EMR releases 7.2 and higher, if you enabled node labels and restricted application processes to `CORE` nodes, Amazon EMR managed scaling scales up core nodes and task nodes if application process demand increases and executor demand increases. Similarly, if you enabled node labels and restricted application processes to `ON_DEMAND` nodes, managed scaling scales up on-demand nodes if application process demand increases and scales up spot nodes if executor demand increases.
- If node labels aren't enabled, application process placement aren't restricted to any node or market type.
- By using node labels, managed scaling can scale up and scale down different instance groups and instance fleets in the same resize operation. For example, in a scenario in which `instance_group1` has `ON_DEMAND` node and `instance_group2` has a `SPOT` node, and node labels are enabled and application processes are restricted to nodes with the `ON_DEMAND` label. Managed scaling will scale down `instance_group1` and scale up `instance_group2` if application process demand decreases and executor demand increases.
- When Amazon EMR experiences a delay in scale-up with the current instance group, clusters that use managed scaling automatically switch to a different task instance group.
- 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`, Amazon EMR first scales core nodes until the core capacity

limit is reached. For the remaining capacity, Amazon 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

- Similar to the scale-up strategy, Amazon EMR removes nodes based on node labels. For more information about node labels, see Understand node types: primary, core, and task nodes.
- If you haven't enabled node labels, managed scaling removes task nodes and then removes core nodes until it achieves the desired scale-down target capacity. Managed scaling never scales down the cluster below the minimum constraints specified in the managed scaling policy.
- Amazon EMR versions 5.34.0 and higher, and Amazon EMR versions 6.4.0 and higher, support Spark shuffle data awareness, which prevents an instance from scaling down while Managed Scaling is aware of existing shuffle data. For more information on shuffle operations, see the Spark Programming Guide. Managed Scaling makes best effort to prevent scaling-down nodes with shuffle data from the current and previous stage of any active Spark application, up to a maximum of 30 minutes. This helps minimize unintended shuffle data loss, avoiding the need for job re-attempts and recomputation of intermediate data. However, prevention of shuffle data loss is not guaranteed. For guaranteed protection, we recommend shuffle awareness on clusters with release label 7.4 or higher. See below for how to set up guaranteed shuffle protection.
- Managed scaling first removes task nodes and then removes core nodes until it achieves the desired scale-down target capacity. The cluster never scales below the minimum constraints specified in the managed scaling policy.
- For clusters that are launched with Amazon EMR 5.x releases 5.34.0 and higher, and 6.x releases 6.4.0 and higher, Amazon EMR-managed scaling doesn’t scale down nodes that have `ApplicationMaster` for Apache Spark running on them. This minimizes job failures and retries, which helps to improve job performance and reduce costs. To confirm which nodes in your cluster are running `ApplicationMaster`, visit the Spark History Server and filter for the driver under the **Executors** tab of your Spark application ID.
- While the intelligent scaling with EMR Managed Scaling minimizes shuffle data loss for Spark, there can be instances when transient shuffle data might be not be protected during a scale-down. To provide enhanced resiliency of shuffle data during scale-down, we recommend enabling **Graceful Decommissioning for Shuffle Data** in YARN. When **Graceful Decommissioning for Shuffle Data** is enabled in YARN, nodes selected for scale-down that have shuffle data will enter the **Decommissioning** state and continue to serve shuffle files. The YARN

ResourceManager waits until nodes report no shuffle files present before removing the nodes from the cluster.

- Amazon EMR version 6.11.0 and higher support Yarn-based graceful decommissioning for **Hive** shuffle data for both the Tez and MapReduce Shuffle Handlers.
  - Enable Graceful Decommissioning for Shuffle Data by setting `yarn.resourcemanager.decommissioning-nodes-watcher.wait-for-shuffle-data` to `true`.
- Amazon EMR version 7.4.0 and higher support Yarn-based graceful decommissioning for Spark shuffle data when the external shuffle service is enabled (enabled by default in EMR on EC2).
  - The default behavior of the Spark external shuffle service, when running Spark on Yarn, is for the Yarn NodeManager to remove application shuffle files at time of application termination. This may have an impact on the speed of node decommissioning and compute utilization. For long running applications, consider setting `spark.shuffle.service.removeShuffle` to `true` to remove shuffle files no longer in use to enable faster decommissioning of nodes with no active shuffle data.
  - If either the `yarn.nodemanager.shuffledata-monitor.interval-ms` flag or the `spark.dynamicAllocation.executorIdleTimeout` has been changed from the default values, ensure that the condition `spark.dynamicAllocation.executorIdleTimeout > yarn.nodemanager.shuffledata-monitor.interval-ms` remains `true` by updating the necessary flag.

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.

This scenario isn't applicable if you use managed scaling with node labels and restrict your application processes to only run on `CORE` nodes, because managed scaling scales task nodes to accommodate executor demand.

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. When you use managed scaling with node labels and restrict your applicati on proccesses to `ON_DEMAND` nodes, the cluster will scale 1 to 20 instances
Instance fleets Core: 1 On-Demand Task: 1 On-Demand and 1 Spot	UnitType: InstanceFleetUnits `MinimumCapacityUnits` : 1 `MaximumCapacityUnits` : 20 `MaximumOnDemandCapacityUnits` : 20 `MaximumCoreCapacityUnits` : 20	

Cluster initial state	Scaling parameters	Scaling behavior
		or instance fleet units on `CORE` nodes using the `On-Demand` or `Spot` type, depending on the type of demand.

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 capa city 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	

Cluster initial state	Scaling parameters	Scaling behavior
	`MaximumCoreCapacityUnits :2`	When you use managed scaling with node labels and restrict your application proccesses to ON_DEMAND nodes, the cluster will keep core nodes steady at 2 and only scale task nodes between 0 to 18 instances or instance fleet units that use the `On-demand` or `Spot` type, depending on the type of demand.

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.

This scenario is not applicable if you use managed scaling with node labels and restrict your application processes to only run on `ON_DEMAND` nodes, because managed scaling scales `Spot` nodes to accommodate executor demand.

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. When you use managed scaling with node labels and restrict your application proccesses to `CORE` nodes, the
Instance fleets Core: 1 On-Demand Task: 1 On-Demand	`UnitType`: InstanceFleetUnits `MinimumCapacityUnits` : 1 `MaximumCapacityUnits` : 20 `MaximumOnDemandCapacityUnits` : 20 `MaximumCoreCapacityUnits` : 12	

Cluster initial state	Scaling parameters	Scaling behavior
		cluster scales between 1 to 20 instances or instance fleet units on `CORE` nodes or `task` nodes using the `ON_DEMAND` type, depending on the type of demand. Scaling on core nodes will not exceed 12 instances or instance fleet units.

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, Amazon EMR managed scaling creates a task group using Spot Instances when needed.

This scenario isn't applicable if you use managed scaling with node labels and restrict your application processes to only run on `ON_DEMAND` nodes, because managed scaling scales `ON_DEMAND` nodes to accommodate application process demand.

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. When you use managed scaling with node labels and restrict your applicati on proccesses to `CORE` nodes, the cluster scales between 1 to 20 instances or instance fleet units on `CORE` or `TASK` nodes using Spot, depending on the type of demand. Amazon EMR
Instance fleets Core: 1 Spot Task: 1 Spot	`UnitType`: InstanceFleetUnits `MinimumCapacityUnits` : 1 `MaximumCapacityUnits` : 20 `MaximumOnDemandCapacityUnits` : 0	

Cluster initial state	Scaling parameters	Scaling behavior
		doesn't scale using the `ON_DEMAND` type.

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.

This scenario isn't applicable if you use managed scaling with node labels and restrict your application processes to only run on `ON_DEMAND` nodes or `CORE` nodes.

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	Scale up to 6 On-Demand units on the core node since there is already 1 On-Demand unit on the task

Cluster initial state	Scaling parameters	Scaling behavior
	`MaximumCoreCapacityUnits : 7`	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`	

Scenario 6: Scale `CORE` instances for application process demand and `TASK` instances for executor demand.

This scenario is only applicable if you use managed scaling with node labels and restrict application processes to only run on `CORE` nodes.

To scale `CORE` nodes based on application process demand and `TASK` nodes based on executor demand, you must set the following configurations at cluster launch:

- `yarn.node-labels.enabled:true`
- `yarn.node-labels.am.default-node-label-expression: 'CORE'`

If you don't specify the `ON_DEMAND` limit and the maximum `CORE` node parameters, both parameters default to the maximum boundary.

If the maximum `ON_DEMAND` node is less than the maximum boundary, managed scaling uses the maximum `ON_DEMAND` node parameter to split capacity allocation between `ON_DEMAND` and `SPOT` nodes. If you set the the maximum `CORE` node parameter to less than or equal to the minimum capacity parameter, `CORE` nodes remain static at the maximum core capacity.

The following examples demonstrate the scenario of scaling CORE instances based on application process demand and TASK instances based on executor demand.

Cluster initial state	Scaling parameters	Scaling behavior
Instance groups Core: 1 On-Demand Task: 1 On-Demand	`UnitType`: Instances `MinimumCapacityUnits` : 1 `MaximumCapacityUnits` : 20 `MaximumOnDemandCapacityUnits` : 10 `MaximumCoreCapacityUnits` : 20	Scales `CORE` nodes between 1 and 20 nodes based on the cluster's application process demand using the On-Demand or Spot market type. Scales `TASK` nodes based on executor demand and remaining available capacity after Amazon EMR allocates `CORE` nodes. The sum of requested `CORE` and `TASK` nodes won't exceed the `maximumCapacity` of 20. The sum of requested on-demand core nodes and on-demand task nodes won't
Instance fleets Core: 1 On-Demand Task: 1 On-Demand	`UnitType`: InstanceFleetUnits `MinimumCapacityUnits` : 1 `MaximumCapacityUnits` : 20 `MaximumOnDemandCapacityUnits` : 10 `MaximumCoreCapacityUnits` : 20	

Cluster initial state	Scaling parameters	Scaling behavior
		exceed the `maximumOnDemandCapacity` of 10. Additional core or task nodes use the Spot market type.

Scenario 7: Scale `ON_DEMAND` instances for application process demand and `SPOT` instances for executor demand.

This scenario is only applicable if you use managed scaling with node labels and restrict application processes to only run on `ON_DEMAND` nodes.

To scale `ON_DEMAND` nodes based on application process demand and `SPOT` nodes based on executor demand, you must set the following configurations at cluster launch:

- `yarn.node-labels.enabled:true`
- `yarn.node-labels.am.default-node-label-expression: 'ON_DEMAND'`

If you don't specify the `ON_DEMAND` limit and the maximum `CORE` node parameters, both parameters default to the maximum boundary.

If the maximum `CORE` node is less than the maximum boundary, managed scaling uses the maximum `CORE` node parameter to split capacity allocation between `CORE` and `TASK` nodes. If you set the the maximum `CORE` node parameter to less than or equal to the minimum capacity parameter, `CORE` nodes remain static at the maximum core capacity.

The following examples demonstrate the scenario of scaling On-Demand Instances based on application process demand and Spot instances based on executor demand.

Cluster initial state	Scaling parameters	Scaling behavior

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` : 10	Scales `ON_DEMAND` nodes between 1 and 20 nodes based on the cluster's application process demand using the `CORE` or `TASK` node type. Scales `SPOT` nodes based on executor demand and remaining available capacity after Amazon EMR allocates `ON_DEMAND` nodes. The sum of requested `ON_DEMAND` and `SPOT` nodes won't exceed the `maximumCapacity` of 20. The sum of requested on-demand core nodes and spot core nodes
Instance fleets Core: 1 On-Demand Task: 1 On-Demand	`UnitType`: InstanceFleetUnits `MinimumCapacityUnits` : 1 `MaximumCapacityUnits` : 20 `MaximumOnDemandCapacityUnits` : 20 `MaximumCoreCapacityUnits` : 10	

Cluster initial state	Scaling parameters	Scaling behavior
		won't exceed the `maximumCoreCapacity` of 10. Additional on-demand or spot nodes use the `TASK` node type.

Understanding managed scaling metrics in Amazon EMR

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 with 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` •	The target total number of units/nodes/vCPUs in a cluster as determined by managed scaling. Units: *Count*

Metric	Description
`TotalVCPURequested`	
• `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*

Metric	Description
• `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*

Metric	Description
`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*
`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
`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*

The following metrics provide information about resources used by YARN containers and nodes. These metrics from the YARN resource manager offer insights into the resources used by containers and nodes running in the cluster. Comparing these metrics to the previous table's cluster capacity metrics provides a clearer picture of the impact of managed scaling:

Metric	Associated releases	Description
`YarnContainersUsedMemoryGBSeconds`	Available to release label 7.3.0 and higher	The consumed container memory * seconds for the publishing period. **Units:** GB * seconds
`YarnContainersTotalMemoryGBSeconds`	Available to release label 7.3.0 and higher	The total yarn container * seconds for the publishing period. **Units:** GB * seconds
`YarnContainersUsedVCPUSeconds`	Available to release label 7.5.0 and higher	The consumed container VCPU * seconds for the publishing period. **Units:** VCPU * seconds
`YarnContainersTotalVCPUSeconds`	Available to release label 7.5.0 and higher	The total container VCPU * seconds for the publishing period. **Units:** VCPU * seconds

Metric	Associated releases	Description
`YarnNodesUsedMemoryGBSeconds`	Available to release label 7.5.0 and higher	The consumed node memory * seconds for the publishing period. **Units:** GB * seconds
`YarnNodesTotalMemoryGBSeconds`	Available to release label 7.5.0 and higher	The total node memory * seconds for the publishing period. **Units:** GB * seconds
`YarnNodesUsedVCPUSeconds`	Available to release label 7.3.0 and higher	The consumed node VCPU * seconds for the publishing period. **Units:** VCPU * seconds
`YarnNodesTotalVCPUSeconds`	Available to release label 7.3.0 and higher	The total node VCPU * seconds for the publishing period. **Units:** VCPU * seconds

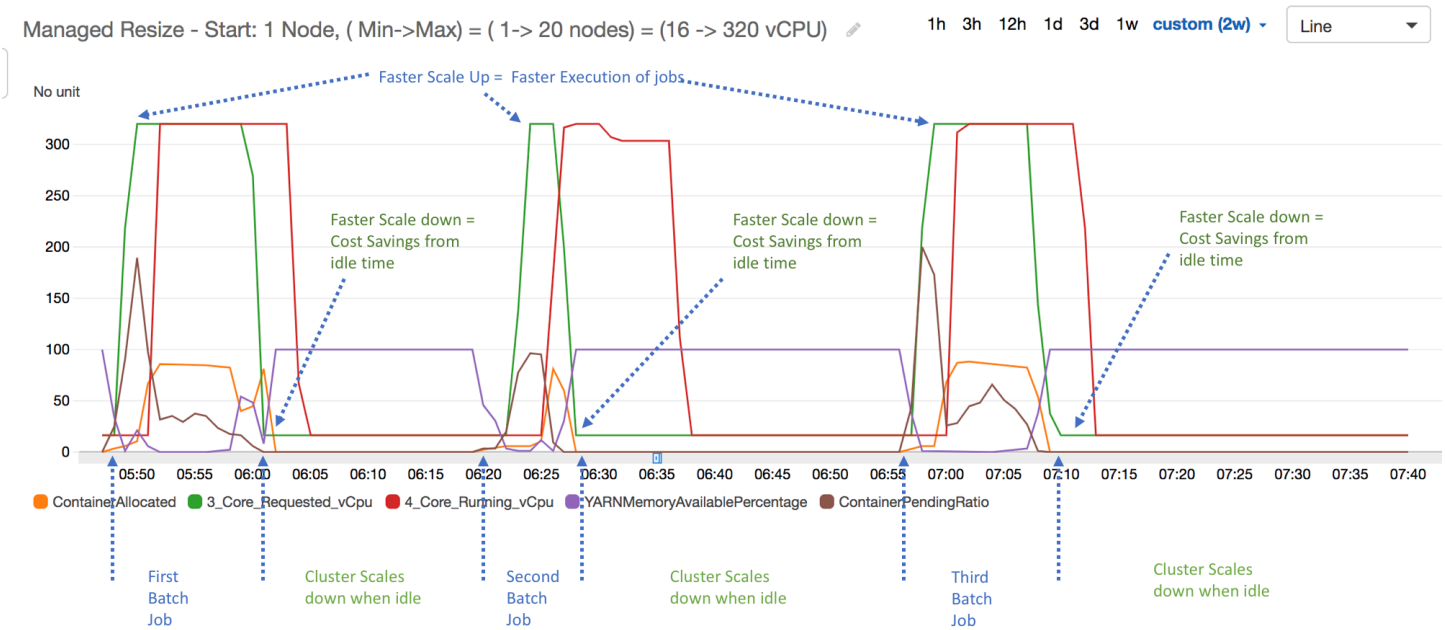
Graphing managed scaling metrics

You can graph metrics to visualize your cluster's workload patterns and corresponding scaling decisions made by Amazon EMR managed scaling as the following steps demonstrate.

To graph managed scaling metrics in the CloudWatch console

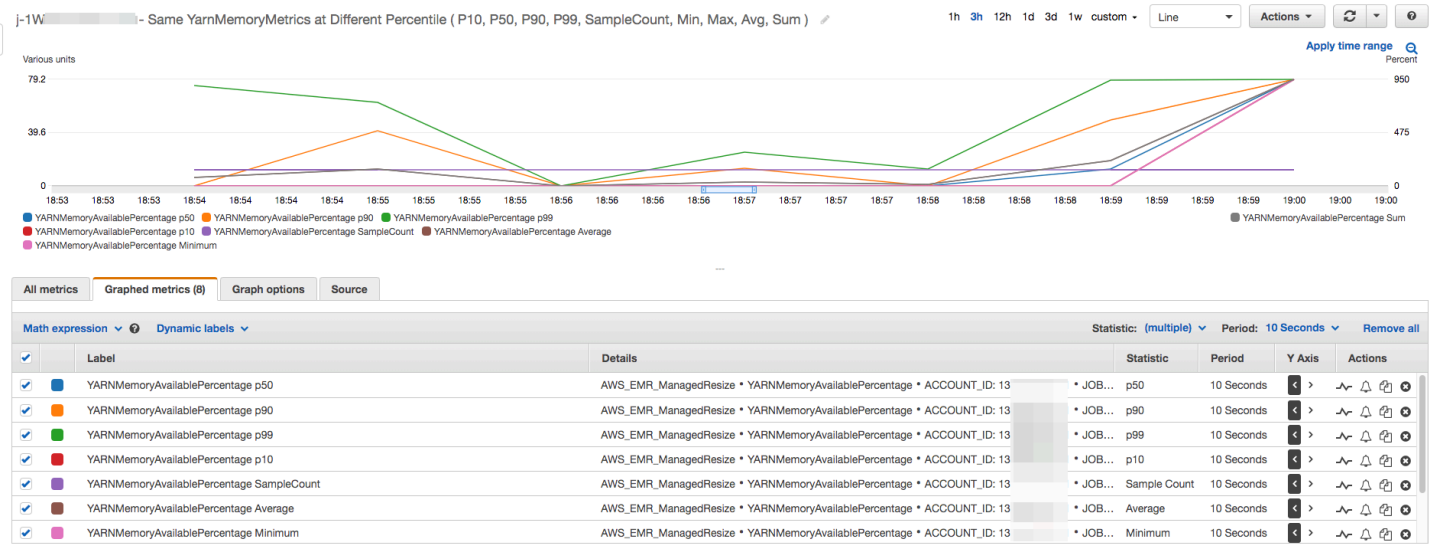
1. Open the CloudWatch console.
2. In the navigation pane, choose **Amazon EMR**. You can search on the cluster identifier of the cluster to monitor.
3. Scroll down to the metric to graph. Open a metric to display the graph.
4. To graph one or more metrics, select the check box next to each metric.

The following example illustrates the Amazon 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`.



Using automatic scaling with a custom policy for instance groups in Amazon EMR

Automatic scaling with a custom policy in Amazon EMR releases 4.0 and higher 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 an Amazon EMR cluster with instance fleets or uniform instance groups.

> **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 primary 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 with 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 with the AWS CLI or the Amazon EMR API, you can specify custom CloudWatch metrics. Custom metrics are not available for selection with the AWS Management Console. When you initially create a scaling policy with 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 with 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 with 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.

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*.

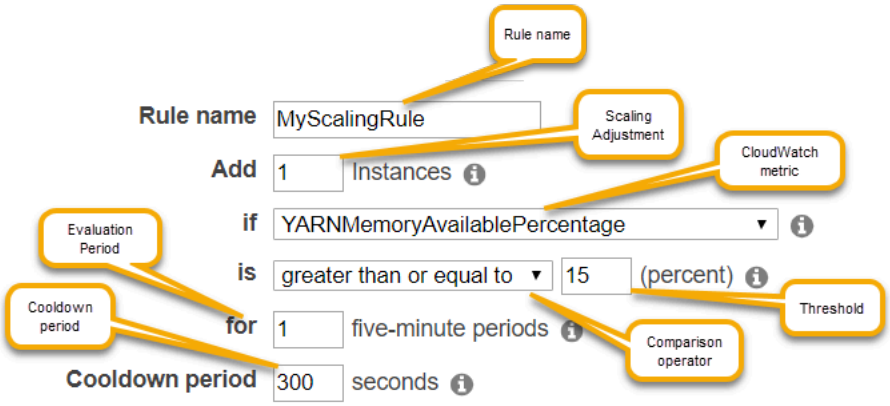
You can configure how a cluster terminates Amazon EC2 instances. You can choose to either terminate at the Amazon EC2 instance-hour boundary for billing, or upon task completion. This setting applies both to automatic scaling and to manual resizing operations. For more information about this configuration, see Cluster scale-down options for Amazon EMR clusters.

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 releases 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 need to use an Amazon EMR release 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 with 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. Navigate to the new Amazon EMR console and select **Switch to the old console** from the side navigation. For more information on what to expect when you switch to the old console, see Using the old console.
2. 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.
3. In **Cluster scaling and provisioning option** 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.
4. 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.
5. 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.
6. 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 with 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
scale-out,Description=Replicates the default scale-out rule in the
 console.,Action={SimpleScalingPolicyConfiguration={AdjustmentType=CHANGE_IN_CAPACITY,ScalingAd
ElasticMapReduce,Period=300,Statistic=AVERAGE,Threshold=15,Unit=PERCENT,Dimensions=[{Key=JobFlo
```

The following command illustrates how to use 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 with 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 resize a running Amazon EMR cluster

You can add and remove instances from core and task instance groups and instance fleets in a running cluster with 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. Applications can use newly provisioned Amazon EC2 instances to host nodes as soon as the instances are available. When instances are removed, Amazon EMR shuts down tasks in a way that does not interrupt jobs and safeguards against data loss. For more information, see Terminate at task completion.

Resize a cluster with the console

You can use the Amazon EMR console to resize a running cluster.

Console

To change the instance count for an existing cluster with the new console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and select the cluster that you want to update. The cluster must be running; you can't resize a provisioning or terminated cluster.
3. On the **Instances** tab on the cluster details page, view the **Instance groups** panel.
4. To resize an existing instance group, select the radio button next to the core or task instance group that you want to resize and then choose **Resize instance group**. Specify the new number of instances for the instance group, then select **Resize**.

   > **Note**
   >
   > If you choose to reduce the size of a running instance group, Amazon EMR will intelligently select the instances to remove from the group for minimal data loss. For more granular control of your resize action, you can select the **ID** for the instance group, choose the instances you want to remove, and then use the **Terminate** option. For more information on intelligent scale-down behavior, see Cluster scale-down options for Amazon EMR clusters.

5. If you want to cancel the resizing action, you can select the radio button for an instance group with the status **Resizing** and then choose **Stop resize** from the list actions.
6. To add one or more task instance groups to your cluster in response to increasing workload, choose **Add task instance group** from the list actions. Choose the Amazon EC2 instance

type, enter the number of instances for the task group, then select **Add task instance group** to return to the **Instance groups** panel for your cluster.

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 with 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 shut down an instance in the core instance group with the AWS CLI or the API. This should be done with caution. Shutting down 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 with the AWS CLI.

To resize a cluster by changing the instance count with the AWS CLI

You can add instances to the core group or task group, and you can remove instances from the task group with 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 with 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
  ```

With 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 with the AWS CLI

With 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 primary instance group and a core instance group. When you use 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 with 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 with the AWS CLI

You can use the AWS CLI to stop a resize with 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 with 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 primary, 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
```

Considerations when reducing cluster size

If you choose to reduce the size of a running cluster, consider the following Amazon EMR behavior and best practices:

- To reduce impact on jobs that are in progress, Amazon EMR intelligently selects the instances to remove. For more information on cluster scale-down behavior, see Terminate at task completion in the Amazon EMR Management Guide.
- When you scale down the size of a cluster, Amazon EMR copies the data from the instances that it removes to the instances that remain. Ensure that there is sufficient storage capacity for this data in the instances that remain in the group.

- Amazon EMR attempts to decommission HDFS on instances in the group. Before you reduce the size of a cluster, we recommend that you minimize HDFS write I/O.
- For the most granular control when you reduce the size of a cluster, you can view the cluster in the console and navigate to the **Instances** tab. Select the **ID** for the instance group that you want to resize. Then use the **Terminate** option for the specific instances that you want to remove.

Configuring provisioning timeouts to control capacity in Amazon EMR

When you use instance fleets, you can configure *provisioning timeouts*. A provisioning timeout instructs Amazon EMR to stop provisioning instance capacity if the cluster exceeds a specified time threshold during cluster launch or cluster scaling operations. The following topics cover how to configure a provisioning timeout for cluster launch and for cluster scale-up operations.

Topics

- Configure provisioning timeouts for cluster launch in Amazon EMR
- Customize a provisioning timeout period for cluster resize in Amazon EMR

Configure provisioning timeouts for cluster launch in Amazon EMR

You can define a timeout period to provision Spot Instances for each fleet in your cluster. If Amazon EMR can't provision Spot capacity, you can choose either to terminate the cluster or to provision On-Demand capacity instead. If the timeout period ends during the cluster resizing process, Amazon EMR cancels unprovisioned Spot requests. Unprovisioned Spot instances aren't transferred to On-Demand capacity.

Perform the following steps to customize a provisioning timeout period for cluster launch with the Amazon EMR console.

Console

To configure the provisioning timeout when you create a cluster with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose **Create cluster**.
3. On the **Create Cluster** page, navigate to **Cluster configuration** and select **Instance Fleets**.

4. Under **Cluster scaling and provisioning option**, specify the Spot size for your core and task fleets.
5. Under **Spot timeout configuration**, select either **Terminate cluster after Spot timeout** or **Switch to On-Demand after Spot timeout**. Then, specify the timeout period for provisioning Spot Instances. The default value is 1 hour.
6. Choose any other options that apply for your cluster.
7. To launch your cluster with the configured timeout, choose **Create cluster**.

AWS CLI

To specify a provisioning timeout with the `create-cluster` command

```
aws emr create-cluster \
--release-label emr-5.35.0 \
--service-role EMR_DefaultRole \
--ec2-attributes '{"InstanceProfile":"EMR_EC2_DefaultRole","SubnetIds":["subnet-
XXXXX"]}' \
--instance-fleets
 '[{"InstanceFleetType":"MASTER","TargetOnDemandCapacity":1,"TargetSpotCapacity":0,"LaunchSp
{"OnDemandSpecification":{"AllocationStrategy":"lowest-
price"}},"InstanceTypeConfigs":[{"WeightedCapacity":1,"EbsConfiguration":
{"EbsBlockDeviceConfigs":[{"VolumeSpecification":
{"SizeInGB":32,"VolumeType":"gp2"},"VolumesPerInstance":2}]},"BidPriceAsPercentageOfOnDemand
 - 1"},
{"InstanceFleetType":"CORE","TargetOnDemandCapacity":1,"TargetSpotCapacity":1,"LaunchSpecifi
{"SpotSpecification":
{"TimeoutDurationMinutes":120,"TimeoutAction":"SWITCH_TO_ON_DEMAND"},"OnDemandSpecification"
{"AllocationStrategy":"lowest-price"}},"InstanceTypeConfigs":
[{"WeightedCapacity":1,"EbsConfiguration":{"EbsBlockDeviceConfigs":
[{"VolumeSpecification":
{"SizeInGB":32,"VolumeType":"gp2"},"VolumesPerInstance":2}]},"BidPriceAsPercentageOfOnDemand
 - 2"}]'
```

Customize a provisioning timeout period for cluster resize in Amazon EMR

You can define a timeout period for provisioning Spot Instances for each fleet in your cluster. If Amazon EMR can't provision the Spot capacity, it cancels the resize request and stops its attempts to provision additional Spot capacity. When you create a cluster, you can configure the timeout. For a running cluster, you can add or update a timeout.

When the timeout period expires, Amazon EMR automatically sends events to an Amazon CloudWatch Events stream. With CloudWatch, you can create rules that match events according to a specified pattern, and then route the events to targets to take action. For example, you might configure a rule to send an email notification. For more information on how to create rules, see Creating rules for Amazon EMR events with CloudWatch. For more information about different event details, see Instance fleet state-change events.

Examples of provisioning timeouts for cluster resize

Specify a provisioning timeout for resize with the AWS CLI

The following example uses the `create-cluster` command to add a provisioning timeout for resize.

```
aws emr create-cluster \
--release-label emr-5.35.0 \
--service-role EMR_DefaultRole \
--ec2-attributes '{"InstanceProfile":"EMR_EC2_DefaultRole","SubnetIds":["subnet-
XXXXX"]}' \
--instance-fleets
 '[{"InstanceFleetType":"MASTER","TargetOnDemandCapacity":1,"TargetSpotCapacity":0,"InstanceTyp
[{"WeightedCapacity":1,"EbsConfiguration":{"EbsBlockDeviceConfigs":
[{"VolumeSpecification":
{"SizeInGB":32,"VolumeType":"gp2"},"VolumesPerInstance":2}]},"BidPriceAsPercentageOfOnDemandPri
 - 1"},
{"InstanceFleetType":"CORE","TargetOnDemandCapacity":1,"TargetSpotCapacity":1,"LaunchSpecificat
{"SpotSpecification":
{"TimeoutDurationMinutes":120,"TimeoutAction":"SWITCH_TO_ON_DEMAND"},"OnDemandSpecification":
{"AllocationStrategy":"lowest-price"}},"ResizeSpecifications":
{"SpotResizeSpecification":{"TimeoutDurationMinutes":20},"OnDemandResizeSpecification":
{"TimeoutDurationMinutes":25}},"InstanceTypeConfigs":
[{"WeightedCapacity":1,"EbsConfiguration":{"EbsBlockDeviceConfigs":
[{"VolumeSpecification":
{"SizeInGB":32,"VolumeType":"gp2"},"VolumesPerInstance":2}]},"BidPriceAsPercentageOfOnDemandPri
 - 2"}]'
```

The following example uses the `modify-instance-fleet` command to add a provisioning timeout for resize.

```
aws emr modify-instance-fleet \
--cluster-id j-XXXXXXXXXXXXX \
```

```
--instance-fleet '{"InstanceFleetId":"if-XXXXXXXXXXXX","ResizeSpecifications":
{"SpotResizeSpecification":{"TimeoutDurationMinutes":30},"OnDemandResizeSpecification":
{"TimeoutDurationMinutes":60}}}' \
--region us-east-1
```

The following example uses the `add-instance-fleet-`command to add a provisioning timeout for resize.

```
aws emr add-instance-fleet \
--cluster-id j-XXXXXXXXXXXXX \
--instance-fleet
 '{"InstanceFleetType":"TASK","TargetOnDemandCapacity":1,"TargetSpotCapacity":0,"InstanceTypeCo
[{"WeightedCapacity":1,"EbsConfiguration":{"EbsBlockDeviceConfigs":
[{"VolumeSpecification":
{"SizeInGB":32,"VolumeType":"gp2"},"VolumesPerInstance":2}]},"BidPriceAsPercentageOfOnDemandPri
{"SpotResizeSpecification":{"TimeoutDurationMinutes":30},"OnDemandResizeSpecification":
{"TimeoutDurationMinutes":35}}}' \
--region us-east-1
```

Specify a provisioning timeout for resize and launch with the AWS CLI

The following example uses the `create-cluster` command to add a provisioning timeout for resize and launch.

```
aws emr create-cluster \
--release-label emr-5.35.0 \
--service-role EMR_DefaultRole \
--ec2-attributes '{"InstanceProfile":"EMR_EC2_DefaultRole","SubnetIds":["subnet-
XXXXX"]}' \
--instance-fleets
 '[{"InstanceFleetType":"MASTER","TargetOnDemandCapacity":1,"TargetSpotCapacity":0,"LaunchSpeci
{"OnDemandSpecification":{"AllocationStrategy":"lowest-price"}},"InstanceTypeConfigs":
[{"WeightedCapacity":1,"EbsConfiguration":{"EbsBlockDeviceConfigs":
[{"VolumeSpecification":
{"SizeInGB":32,"VolumeType":"gp2"},"VolumesPerInstance":2}]},"BidPriceAsPercentageOfOnDemandPri
 - 1"},
{"InstanceFleetType":"CORE","TargetOnDemandCapacity":1,"TargetSpotCapacity":1,"LaunchSpecificat
{"SpotSpecification":
{"TimeoutDurationMinutes":120,"TimeoutAction":"SWITCH_TO_ON_DEMAND"},"OnDemandSpecification":
{"AllocationStrategy":"lowest-price"}},"ResizeSpecifications":
{"SpotResizeSpecification":{"TimeoutDurationMinutes":20},"OnDemandResizeSpecification":
{"TimeoutDurationMinutes":25}},"InstanceTypeConfigs":
```

```
[{"WeightedCapacity":1,"EbsConfiguration":{"EbsBlockDeviceConfigs":
[{"VolumeSpecification":
{"SizeInGB":32,"VolumeType":"gp2"},"VolumesPerInstance":2}]},"BidPriceAsPercentageOfOnDemandPri
 - 2"}]'
```

Considerations for resize provisioning timeouts

When you configure cluster provisioning timeouts for your instance fleets, consider the following behaviors.

- You can configure provisioning timeouts for both Spot and On-Demand Instances. The minimum provisioning timeout is 5 minutes. The maximum provisioning timeout is 7 days.
- You can only configure provisioning timeouts for an EMR cluster that uses instance fleets. You must configure each core and task fleet separately.
- When you create a cluster, you can configure provisioning timeouts. You can add a timeout or update an existing timeout for a running cluster.
- If you submit multiple resize operations, then Amazon EMR tracks provisioning timeouts for every resize operation. For example, set the provisioning timeout on a cluster to *60* minutes. Then, submit a resize operation *R1* at time *T1*. Submit a second resize operation *R2* at time *T2*. The provisioning timeout for R1 expires at *T1 + 60 minutes*. The provisioning timeout for R2 expires at *T2 + 60 minutes*.
- If you submit a new scale-up resize operation before the timeout expires, Amazon EMR continues its attempt to provision capacity for your EMR cluster.

Cluster scale-down options for Amazon EMR clusters

> **Note**
>
> Scale-down behavior options are no longer supported since Amazon EMR release 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 releases 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 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 attempts to gracefully decommission YARN, HDFS, and other daemons on core and task nodes during a resize down operation without losing data or interrupting jobs. Amazon EMR only reduces instance group size 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.

> **Note**
>
> When graceful decommissioning occurs, there can be data loss. Be sure to back up your data.

> **Important**
>
> It is possible that HDFS data can be permanently lost during the graceful replacement of an unhealthy core instance. We recommend that you always back up your data.

This time is set using a property in the `YARN-site` configuration classification. Using Amazon EMR release 5.12.0 and higher, specify the `YARN.resourcemanager.nodemanager-graceful-decommission-timeout-secs` property. Using earlier Amazon EMR releases, 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 reduction to wait longer. For more information, see Graceful Decommission of YARN nodes in the Apache Hadoop documentation.

Task node groups

Amazon EMR intelligently selects instances that do not have tasks that are running against any step or application, and removes those instances from a cluster first. If all instances in the cluster are in use, Amazon EMR waits for tasks to complete on an instance before removing it from the cluster. The default wait time is 1 hour. This value can be changed with the `YARN.resourcemanager.decommissioning.timeout` setting. Amazon EMR dynamically uses the new setting. You can set this to an arbitrarily large number to ensure that Amazon EMR doesn't terminate any tasks while reducing the cluster size.

Core node groups

On core nodes, both YARN NodeManager and HDFS DataNode daemons must be decommissioned for the instance group to reduce. For YARN, graceful reduction 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 reduction 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 reduce instance groups. Excessive write I/O might delay completion of the resize operation.

Another limit is the default replication factor, `dfs.replication` inside `/etc/hadoop/conf/hdfs-site`. When it creates a cluster, 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.

Warning

1. Setting `dfs.replication` to 1 on clusters with fewer than four nodes can lead to HDFS data loss if a single node goes down. We recommend you use a cluster with at least four core nodes for production workloads.
2. Amazon EMR will not allow clusters to scale core nodes below `dfs.replication`. For example, if `dfs.replication = 2`, the minimum number of core nodes is 2.
3. When you use Managed Scaling, Auto-scaling, or choose to manually resize your cluster, we recommend that you to set `dfs.replication` to 2 or higher.

Graceful reduction doesn't let you reduce core nodes below the HDFS replication factor. This is to allow HDFS to close files due insufficient replicas. To circumvent this limit, 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 5.10.0 and higher. The following scale-down behavior options only appear in the Amazon EMR console for releases 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.

Console

To configure scale-down behavior with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**, and then choose **Create cluster**.
3. In the **Cluster scaling and provisioning option** section, choose **Use custom automatic scaling** . Under **Custom automatic scaling policies**, choose the **plus action button** to add

Scale in policies. We recommend that you add both **Scale in** and **Scale out** policies. Adding in only one set of policies means Amazon EMR will only perform one-way scaling, and you have to manually perform the other actions.

4. Choose any other options that apply to your cluster.
5. To launch your cluster, choose **Create cluster**.

AWS CLI

To configure scale-down behavior with the AWS CLI

- Use the `--scale-down-behavior` option to specify either `TERMINATE_AT_INSTANCE_HOUR` or `TERMINATE_AT_TASK_COMPLETION`.

Terminate an Amazon EMR cluster in the starting, running, or waiting states

This section describes the methods of terminating a cluster. For information about enabling termination protection and auto-terminating clusters, see Control Amazon EMR cluster termination. 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 want to terminate 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 could take from 5 to 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 Clone an Amazon EMR cluster using the console.

> **Important**
>
> Amazon EMR uses the Amazon EMR service role and the `AWSServiceRoleForEMRCleanup` role to clean up cluster resources in your account that you no longer use, such as Amazon EC2 instances. You must include actions for the role policies to delete or terminate the resources. Otherwise, Amazon EMR can't perform these cleanup actions, and you might incur costs for unused resources that remain on the cluster.

Terminate a cluster with 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.

Console

To terminate a cluster with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Choose **Clusters**, and then choose the cluster you want to terminate.
3. Under the **Actions** dropdown menu, choose **Terminate cluster** to open the **Terminate cluster** prompt.
4. At the prompt, choose **Terminate**. Depending on the cluster configuration, termination may take 5 to 10 minutes. For more information on how to Amazon EMR clusters, see Terminate an Amazon EMR cluster in the starting, running, or waiting states.

Terminate a cluster with 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 with 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.

Clone an Amazon EMR 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.

Console

To clone a cluster with the console

1. Sign in to the AWS Management Console, and open the Amazon EMR console at https://console.aws.amazon.com/emr.
2. Under **EMR on EC2** in the left navigation pane, choose **Clusters**.
3. *To clone a cluster from the cluster list*
   a. Use the search and filter options to find the cluster that you want to clone in the list view.
   b. Select the check box to the left of the row for the cluster that you want to clone.
   c. The **Clone** option will now be available at the top of the list view. Select **Clone** to initiate the cloning process. If the cluster has steps configured, choose **Include steps** and **Continue** if you want to clone the steps along with the other cluster configurations.
   d. Review the settings for the new cluster that have copied over from the cloned cluster. Adjust the settings if needed. When you are satisfied with the new cluster's configuration, select **Create cluster** to launch the new cluster.
4. *To clone a cluster from a cluster detail page*

a. To navigate to the detail page of the cluster that you want to clone, select its **Cluster ID** from the cluster list view.
b. At the top of the cluster detail page, select **Clone cluster** from the **Actions** menu to initiate the cloning process. If the cluster has steps configured, choose **Include steps** and **Continue** if you want to clone the steps along with the other cluster configurations.
c. Review the settings for the new cluster that have copied over from the cloned cluster. Adjust the settings if needed. When you are satisfied with the new cluster's configuration, select **Create cluster** to launch the new cluster.

Automate recurring Amazon EMR 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

Amazon EMR tutorials

An EMR cluster runs in a complex ecosystem. The following tutorial covers important use cases.

Topics

- Amazon EMR on EC2 – Enhanced Monitoring with CloudWatch using custom metrics and logs
- Monitor Apache Spark applications on Amazon EMR with Amazon CloudWatch
- Monitor Amazon EMR application status with CloudWatch integration

Amazon EMR on EC2 – Enhanced Monitoring with CloudWatch using custom metrics and logs

Overview

Amazon EMR provides powerful, cost-effective big data processing capabilities. To maximize performance and resource utilization, effective monitoring is essential. Amazon CloudWatch offers comprehensive observability for EMR clusters, enabling you to track metrics and logs in real-time. This document outlines how to:

1. Configure the CloudWatch agent to send EMR on EC2 logs to CloudWatch
2. Add custom Hadoop, YARN, and HBase metrics through classifications
3. Monitor metrics through built-in dashboards
4. Track cluster logs via CloudWatch log groups

Prerequisites and Background

By default, Amazon EMR sends basic metrics to CloudWatch every five minutes at no additional cost. With EMR Release 7.0+, you can deploy the CloudWatch Agent to:

- Collect 34 additional detailed metrics at one-minute intervals (additional charges apply)
- Gather metrics from all cluster nodes
- Aggregate data on the primary node before sending to CloudWatch
- Access metrics through the EMR console's Monitoring tab or CloudWatch Console

EMR 7.1 extends these capabilities, allowing you to configure the agent to capture specialized metrics from Hadoop, YARN, and HBase components. For environments using Prometheus, metrics can be forwarded to Amazon Managed Service for Prometheus.

CloudWatch Agent Configuration for Logs

To capture EMR logs in CloudWatch, create a *cloudwatch-config.json* file that defines which log files to collect:

cloudwatch-config.json

```
{
  "agent": {"metrics_collection_interval":60,"logfile":"/var/log/
emr-cluster-metrics/amazon-cloudwatch-agent/amazon-cloudwatch-
agent.log","run_as_user":"****","omit_hostname":true},
  "logs": {
    "logs_collected": {
      "files": {
        "collect_list": [
          {
            "file_path": "/mnt/var/log/hadoop-yarn/hadoop-yarn-resourcemanager-*",
            "log_group_name": "/emr/yarn/resourcemnger",
            "log_stream_name": "{instance_id}",
            "publish_multi_logs" : true
          },
          {
            "file_path": "/var/log/hadoop-hdfs/hadoop-hdfs-namenode-*",
            "log_group_name": "/emr/hdfs/namenode",
            "log_stream_name": "{instance_id}",
            "publish_multi_logs" : true
          }
        ]
      }
    }
  }
}
```

Bootstrap Script for CloudWatch Agent Configuration

To apply your custom CloudWatch configuration to EMR nodes, create a bootstrap script that will restart the CloudWatch agent with your settings. This script ensures the agent runs with your specific log collection parameters after cluster provisioning.

Creating the Bootstrap Script

Create a file named *cloudwatch-agent-bootstrap.sh* with the following content:

```
#!/bin/bash
set -xe

EMR_SECONDARY_BA_SCRIPT=$(cat << 'EOF'
while true; do
NODEPROVISIONSTATE=$(sed -n '/localInstance [{]/,/[}]/ {/nodeProvisionCheckinRecord
 [{]/,/[}]/ {/status:/ p}}' /emr/instance-controller/lib/info/job-flow-state.txt | awk
 '{ print $2 }')

if [ "$NODEPROVISIONSTATE" == "SUCCESSFUL" ]; then
sleep 10
echo "Running my post provision bootstrap"
NODETYPE=$(cat /mnt/var/lib/instance-controller/extraInstanceData.json | jq -r
 '.instanceRole' | awk '{print tolower($0)}')

# Copy config file on the instance
sudo aws s3 cp s3://amzn-s3-demo-bucket1>/cloudwatch-config.json /etc/emr-cluster-
metrics/amazon-cloudwatch-agent/conf/emr-amazon-cloudwatch-agent.json

# Stop the current agent
sudo /usr/bin/amazon-cloudwatch-agent-ctl -a stop

# Start the agent with the created config file
sudo /usr/bin/amazon-cloudwatch-agent-ctl -a fetch-config -s -m ec2 -c file:/etc/emr-
cluster-metrics/amazon-cloudwatch-agent/conf/emr-amazon-cloudwatch-agent.json

# Status CW Agent
echo "Status CW Agent"
sudo /usr/bin/amazon-cloudwatch-agent-ctl -m ec2 -a status

exit
fi

sleep 10
done
EOF
)

echo "${EMR_SECONDARY_BA_SCRIPT}" | tee -a /tmp/emr-secondary-ba.sh
chmod u+x /tmp/emr-secondary-ba.sh
```

```
/tmp/emr-secondary-ba.sh > /tmp/emr-secondary-ba.log 2>&1 &
exit 0
```

Important Configuration Note

> **Important**
>
> Before uploading the script, replace **`<amzn-s3-demo-bucket1>`** with the actual name of your S3 bucket where you stored the *cloudwatch-config.json* file from the previous step. This ensures the bootstrap script can retrieve your configuration file during cluster initialization.

This bootstrap script will:

- Wait for node provisioning to complete
- Download your custom CloudWatch configuration
- Stop any running CloudWatch agent
- Restart the agent with your specific configuration
- Log the agent's status for troubleshooting

Custom Metric Classifications for Hadoop, YARN, and HBase

In addition to the default CloudWatch metrics, you can enhance your monitoring capabilities by configuring custom application-specific metrics for your EMR cluster components. Amazon EMR's configuration API provides a flexible way to define exactly which metrics you want to collect.

Configuring Custom Metrics

You can implement custom metric collection in two ways:

- During cluster creation for new clusters
- As a reconfiguration for existing clusters through the EMR console

Creating a Classification File

The classification file defines which specific component metrics should be collected from your cluster. Below is a sample structure for collecting custom Hadoop metrics:

```
[
  {
    "Classification": "emr-metrics",
    "Configurations": [
      {
        "Classification": "emr-hadoop-hdfs-datanode-metrics",
        "Properties": {
          "Hadoop:service=DataNode,name=DataNodeActivity-*":
 "DatanodeNetworkErrors,TotalReadTime,TotalWriteTime,BytesRead,BytesWritten,RemoteBytesRead,Rem
          "otel.metric.export.interval": "30000"
        }
      },
      {
        "Classification": "emr-hadoop-yarn-nodemanager-metrics",
        "Properties": {
          "Hadoop:service=NodeManager,name=JvmMetrics":
 "MemNonHeapUsedM,MemNonHeapCommittedM,MemNonHeapMaxM,MemHeapUsedM,MemHeapCommittedM,MemHeapMax
          "Hadoop:service=NodeManager,name=NodeManagerMetrics":
 "ContainerCpuUtilization,NodeCpuUtilization,ContainersCompleted,ContainersFailed,ContainersKil
          "otel.metric.export.interval": "20000"
        }
      }
    ],
    "Properties": {}
  }
]
```

Implementation Steps

1. Create a JSON file with your desired metric classifications.
2. Customize the metrics based on your monitoring requirements.
3. Save the file and upload it to your S3 bucket.
4. Reference this file when creating a new cluster or reconfiguring an existing one.

Best Practices

- Only collect metrics that provide meaningful insights for your workloads.
- Consider the metrics collection interval based on your monitoring needs.
- Review AWS documentation for the complete list of available metrics for each component.

- Group related metrics within the same classification for better organization.

This approach allows you to focus your monitoring on the most critical metrics for your specific EMR applications, giving you deeper visibility into cluster performance.

Deploying an EMR Cluster with CloudWatch Integration

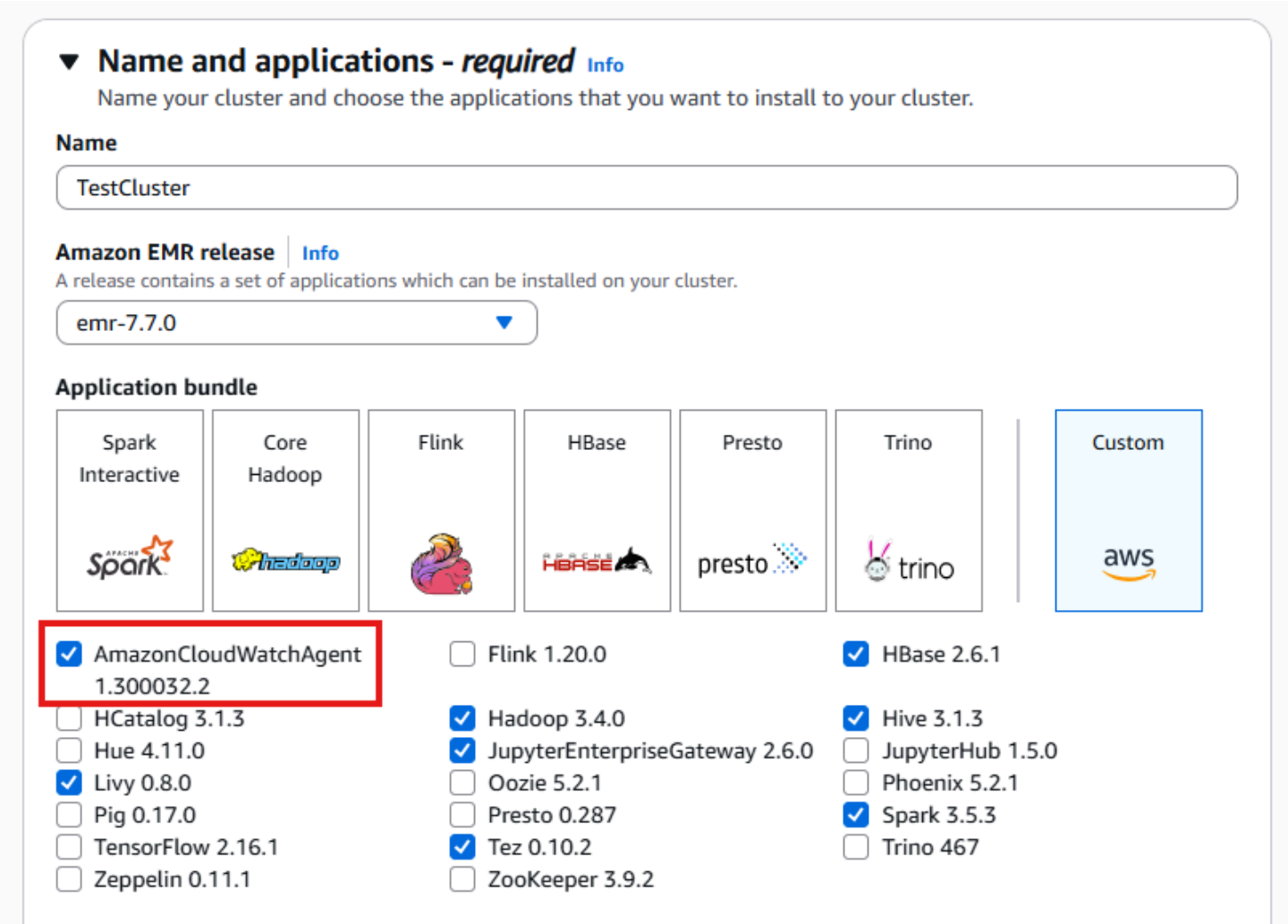
Follow these steps to create an Amazon EMR cluster that automatically sends logs and custom metrics to CloudWatch:

Step 1: Enable the CloudWatch Agent

When creating your EMR cluster through the AWS Management Console:

1. Navigate to the **Applications** section during cluster creation.
2. Select the checkboxes for your primary applications (Hadoop, Spark, etc.).
3. Scroll to find and select the **Amazon CloudWatch Agent** option.
4. This enables the agent on your cluster, which is essential for collecting enhanced metrics and logs.

The CloudWatch Agent will be installed on all nodes in your cluster, allowing it to gather system and application metrics at the configured intervals.



Creating a cluster and showing the available bundles.

> **Note**
>
> The CloudWatch Agent is available in EMR release 7.0 and later. Enabling this component is required for the custom metric collection and log forwarding described in this guide.

Step 2: Add the Bootstrap Action for Log Collection

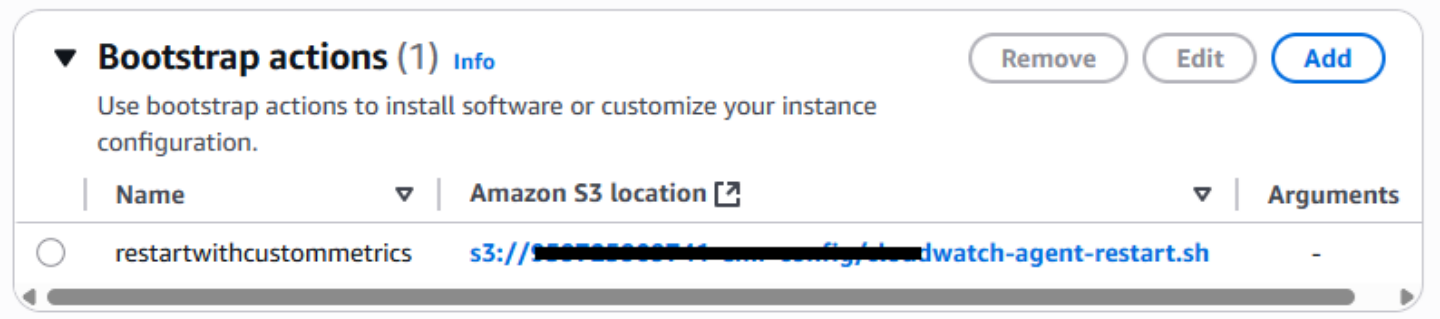
To configure the CloudWatch agent to collect and forward specific log files to CloudWatch:

1. In the EMR cluster creation wizard, navigate to the **Bootstrap Actions** section
2. Click **Add bootstrap action**
3. Select **Custom action** from the dropdown menu

4. Provide a name for your bootstrap action (e.g., **Configure CloudWatch Agent**)
5. In the **Script location** field, enter the S3 path to your cloudwatch-agent-bootstrap.sh script (e.g., s3://your-bucket-name/cloudwatch-agent-bootstrap.sh)
6. Click **Add** to save the bootstrap action

This bootstrap action will execute during cluster startup, ensuring that the CloudWatchagent is properly configured with your custom settings to collect and forward the log files specified in your configuration file.

The agent will automatically begin collecting logs once the nodes are provisioned, providing near real-time visibility into your cluster operations through CloudWatch Logs.



Using bootstrap actions.

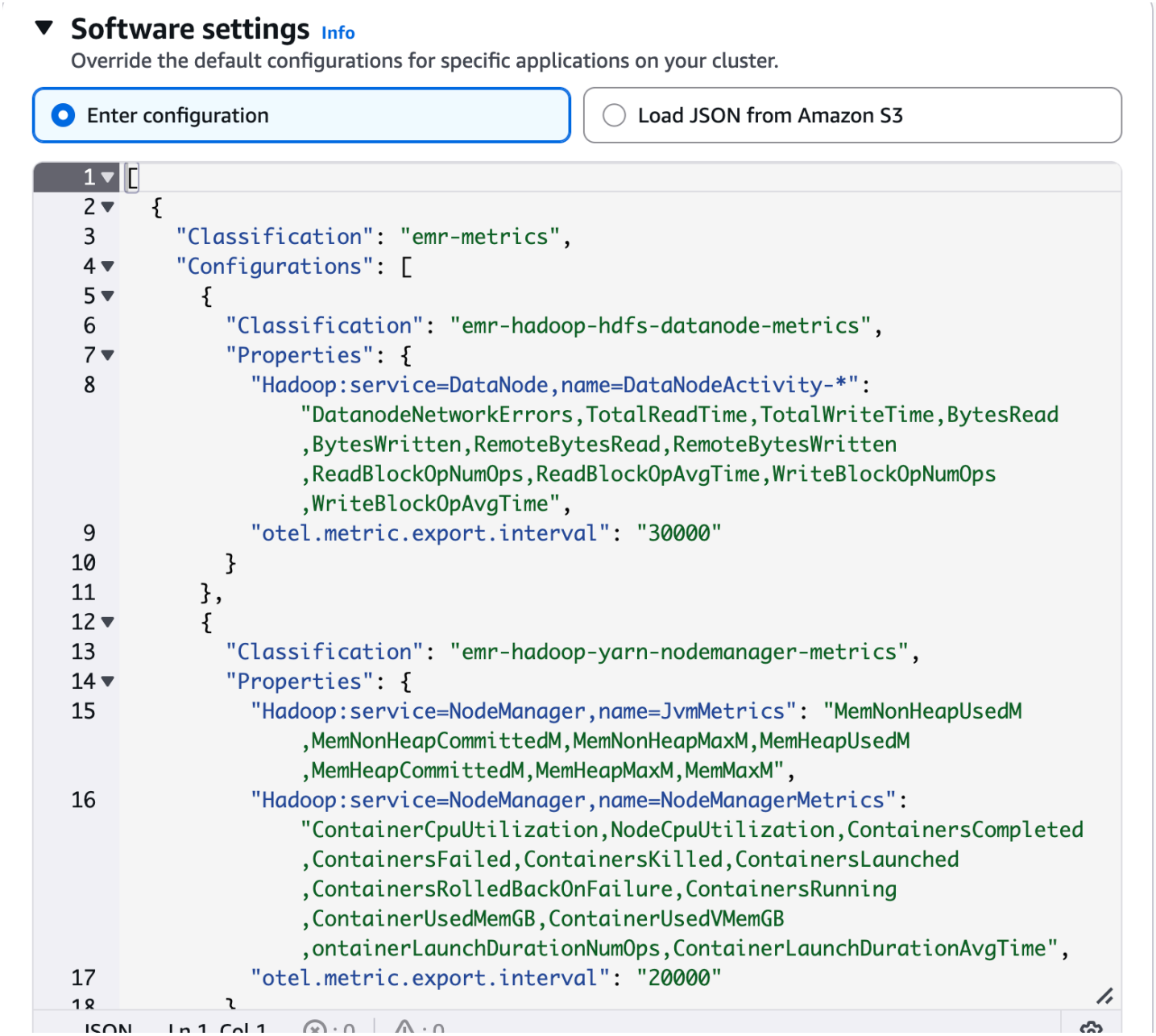
Step 3: Configure Custom Metrics Collection

To enable the collection of custom Hadoop, YARN, or HBase metrics beyond the default set:

1. In the EMR cluster creation wizard, navigate to the **Configurations** section.
2. Click the **Edit configurations** button to expand configuration options.
3. Select **Load JSON from Amazon S3** option from the configuration method dropdown.
4. Enter the S3 URI path to your custom metrics classification file (e.g., s3://amzn-s3-demo-bucket1/emr-metrics-classification.json).
5. Click **Load** to parse the configuration.
6. Verify that the configuration appears correctly in the console interface.
7. Click **Save changes** to apply these metric configurations to your cluster.

This step instructs the CloudWatch agent to collect the specific component metrics defined in your classification file. The metrics will be gathered at the intervals specified in your configuration and published to CloudWatch, where they can be visualized and analyzed.

Custom metrics provide deeper insights into your cluster's performance characteristics, allowing for more precise monitoring and troubleshooting of your EMR applications.



Override the default configurations.

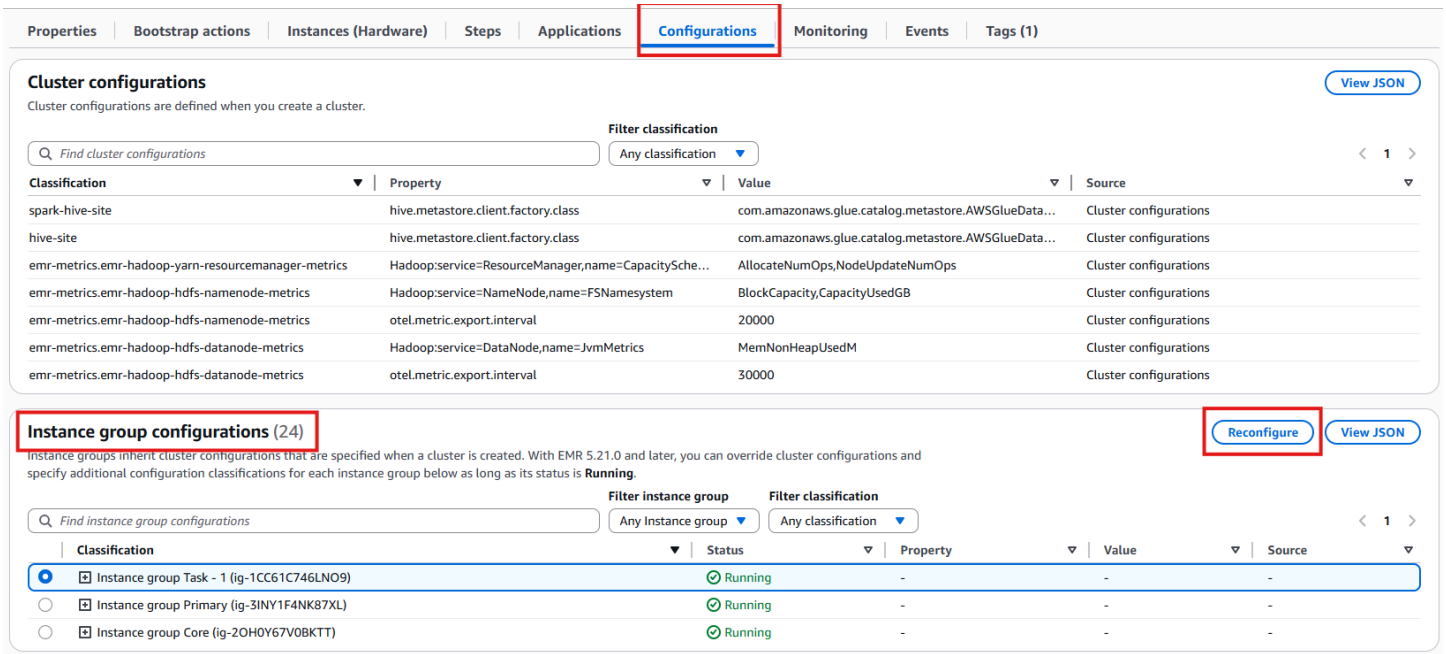
Updating Metrics Configuration for Running Clusters

You can modify the metrics collection settings for an existing EMR cluster without disrupting operations by following these steps:

1. Navigate to your active EMR cluster in the AWS Management Console.
2. Select the **Configurations** tab in the cluster details view.
3. Find the **Instance group configurations** section.
4. Click the **Reconfigure** button to modify settings.
5. Choose **Load JSON from Amazon S3** or directly edit the configuration.
6. Enter your updated metrics classification file location or make changes in the editor.
7. Apply the changes to update the metrics collection behavior.

This reconfiguration capability allows you to fine-tune your monitoring approach as your workload requirements evolve. The CloudWatch agent will automatically adapt to the new configuration, collecting the updated set of metrics without requiring cluster restarts or downtime.

> **Important**
>
> Configuration changes may take several minutes to propagate across all nodes in the cluster. Continue monitoring your CloudWatch dashboards to confirm the new metrics appear as expected.



Instance group configurations.

Validating your CloudWatch integration

After completing the configuration steps, it's time to verify that your monitoring setup is working correctly:

Step 1: Deploy Your EMR Cluster

1. Review all configuration settings for accuracy.
2. Ensure bootstrap actions and classification files are correctly referenced.
3. Click **Create cluster** to launch your EMR environment.
4. Wait for the cluster to reach the **Running** state (typically 5-15 minutes).

Step 2: Execute Test Applications

Submit several test Spark applications to generate meaningful metrics:

- Run a simple Spark job that processes sample data.
- Execute a longer-running analytics task to observe resource utilization.
- Test different application configurations to compare performance metrics.

After your applications complete (or while they're running):

- Navigate to the CloudWatch console.
- Check your configured log groups for application logs.
- Examine the metrics dashboards to observe CPU, memory, and application-specific metrics.
- Verify that custom metrics defined in your classification file appear in CloudWatch.

This validation process confirms that your CloudWatch integration is properly capturing both logs and metrics, providing you with comprehensive visibility into your EMR cluster's performance and application behavior.

Accessing EMR Logs in CloudWatch Log Groups

After your EMR cluster is running and the CloudWatch agent is properly configured, your application and system logs will be available in CloudWatch Logs. Follow these steps to access and analyze them:

Viewing Your Log Groups

1. Navigate to the CloudWatch console in AWS Management Console.
2. Select **Log groups** from the left navigation pane.
3. Look for the log groups created by your configuration, such as:
   - /emr/yarn/resourcemnger for YARN ResourceManager logs.
   - /emr/hdfs/namenode for HDFS NameNode logs.
   - Any additional log groups specified in your configuration file.

Each log group contains log streams organized by instance ID, allowing you to trace logs to specific nodes in your cluster.

Working with Log Data

- *Search Log Data:* Use CloudWatch Logs Insights to perform structured queries across your log groups.
- *Create Metrics:* Extract metrics from log patterns to create custom CloudWatch metrics.
- *Set Alerts:* Configure alarms based on specific error patterns or log frequencies.
- *Export Logs:* Download logs for offline analysis or archiving.

Log Retention

> **Note**
>
> By default, logs are retained for 30 days. You can modify the retention policy for each log group to keep logs for longer periods if needed for compliance or analysis purposes.

CloudWatch Logs provides a centralized location for all your EMR log data, eliminating the need to SSH into individual cluster nodes to troubleshoot issues or analyze application behavior.

Viewing Custom Metrics in the EMR Monitoring Dashboard

After your EMR cluster is running with the CloudWatch agent and custom metrics configuration, you can easily monitor these metrics directly in the EMR console:

Accessing Your Custom Metrics

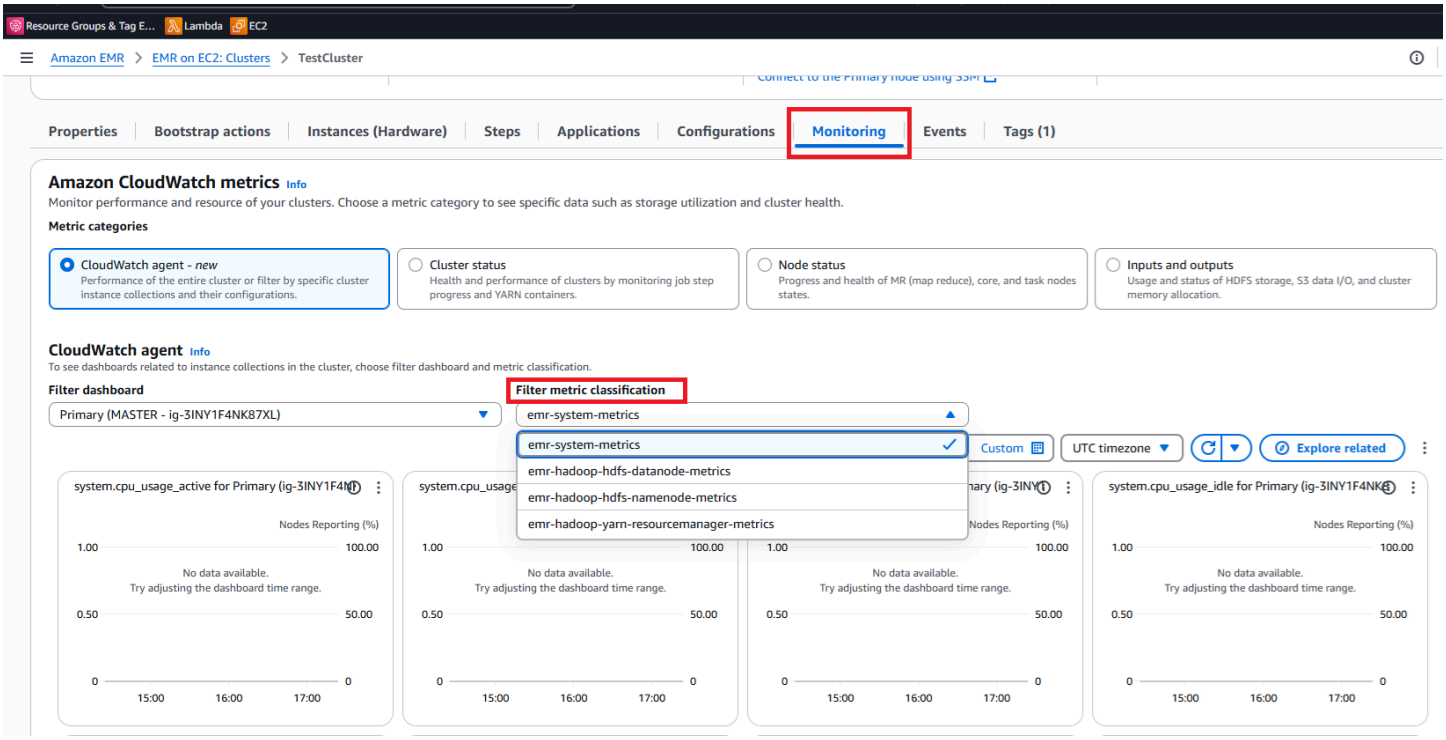
1. Navigate to your EMR cluster in the AWS Management Console.
2. Select the **Monitoring** tab in the cluster details page.
3. Locate the **Filter metric classification** dropdown near the top of the monitoring dashboards.
4. Use this filter to select specific metric categories:
   - Choose **HDFS** to view NameNode and DataNode metrics.
   - Select **YARN** to see ResourceManager and container metrics.
   - Pick **HBase** for HBase-specific performance data.
   - Select custom metric classifications you defined.

The dashboard will dynamically update to display graphs for your selected metrics, showing performance trends over time.

Working with Metric Visualizations

- *Adjust time ranges:* Change the time window to view recent activity or historical trends.
- *Compare metrics:* Display multiple related metrics side-by-side for correlation analysis.
- *Zoom features:* Focus on specific time periods where anomalies or patterns appear.
- *Refresh data:* Update visualizations with the latest metrics data in near real-time.

This integrated monitoring approach allows you to track both standard EMR metrics and your custom metrics in a unified dashboard, making it easier to identify performance issues, resource constraints, or application bottlenecks without leaving the EMR console.



Filtering metrics classification.

Monitor Apache Spark applications on Amazon EMR with Amazon CloudWatch

To improve a Spark application's efficiency, it's essential to monitor its performance and behavior. In the following blog post, a tutorial demonstrates in a step-by-step manner how to publish detailed Spark metrics from Amazon EMR to CloudWatch. This gives you the ability to identify bottlenecks while optimizing resource utilization: Monitor Apache Spark applications on Amazon EMR with Amazon CloudWatch

For more information about CloudWatch, see What is Amazon CloudWatch?.

Monitor Amazon EMR application status with CloudWatch integration

When you integrate CloudWatch with Amazon EMR, you can track critical statuses for applications like HiveServer2. You can publish status to CloudWatch custom metrics and configure alerts for service unavailability.

Specifically, you can create a script to monitor Amazon EMR applications like YARN ResourceManager and HiveServer2 on a primary node. See Publish and monitor an Amazon EMR application status with CloudWatch integration in the *re:Post Knowledge Center* for details on how to configure this use case.

Troubleshoot Amazon EMR clusters

An EMR cluster runs in a complex ecosystem that comprises open-source software, custom application code, and AWS services. When a problem occurs with any of these parts, the cluster might fail or take longer than you expect for it to complete. The following topics can help you identify cluster issues and how to fix them.

Topics

- What tools are available for troubleshooting an Amazon EMR cluster?
- View and restart Amazon EMR and application processes (daemons)
- Collections of common errors in Amazon EMR
- Troubleshoot an Amazon EMR cluster that has failed with an error code
- Troubleshoot a slow Amazon EMR cluster
- Troubleshoot common issues when using Amazon EMR with AWS Lake Formation

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 Amazon EMR cluster logging and debugging and Step 5: Test the Amazon EMR cluster step by step.

What tools are available for troubleshooting an Amazon EMR cluster?

To identify and fix cluster errors, you can use the tools described on this page. You might need to initialize some of the tools when you launch the cluster. Other tools are available for every cluster by default.

Topics

- View EMR cluster details
- View EMR cluster error details
- Run scripts and configure Amazon EMR processes
- View log files

- Monitor EMR cluster performance

View EMR 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 Amazon EMR cluster status and details.

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 AWS Region. The list displays all active and terminated clusters that you launched in the past two months. 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 useful to troubleshoot clusters. 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 Amazon EMR application history. This feature is available only for Amazon EMR releases 5.8.0 and higher.

Amazon EMR command line interface

You can locate details about a cluster from the AWS CLI with the `--describe` argument.

Amazon EMR API

You can locate details about a cluster from the API using the `DescribeJobFlows` action.

View EMR cluster error details

When an EMR cluster terminates with an error, the `DescribeCluster` and `ListClusters` APIs return an error code and an error message. For select cluster errors, the `ErrorDetail` data array can help you troubleshoot the failure.

For a list of error codes that include `ErrorDetail` data, see Error codes with ErrorDetail information in Amazon EMR.

> **Note**
>
> We continuously refine our error messages so that you receive the most recent and pertinent information. We don't recommend that you parse the text from `ErrorMessage` because this text is subject to change.

Run scripts and configure Amazon EMR 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 View and restart Amazon EMR and application processes (daemons).

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.

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 that you specified when you launched the cluster. For more information, see Configure Amazon EMR cluster logging and debugging.

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.

Monitor EMR 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.

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 Monitoring Amazon EMR metrics with CloudWatch.

View and restart Amazon EMR and application processes (daemons)

When you troubleshoot a cluster, you may want to list running processes. You may also want to stop or restart processesS. 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 an Amazon EMR cluster.

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.

Collections of common errors in Amazon EMR

Sometimes, clusters fail or are slow to process data. The following sections list common cluster issues. Errors include bootstrap failures and validation errors, with suggestions on how to resolve them.

Topics

- Error codes with ErrorDetail information in Amazon EMR
- Resource errors during Amazon EMR cluster operations
- Cluster input and output errors during Amazon EMR operations
- Permissions errors during Amazon EMR cluster operations
- Hive cluster errors
- VPC errors during Amazon EMR cluster operations
- Streaming Amazon EMR cluster errors
- Amazon EMR: Custom JAR cluster errors
- Amazon EMR AWS GovCloud (US-West) errors
- Find a missing cluster

Error codes with ErrorDetail information in Amazon EMR

When an EMR cluster terminates with an error, the `DescribeCluster` and `ListClusters` APIs return an error code and an error message. For some cluster errors, the `ErrorDetail` data array can help you troubleshoot the failure.

Errors that include an `ErrorDetail` array provide the following details:

`ErrorCode`

A unique error code that you can use for programmatic access.

`ErrorData`

A list of identifiers in key-value pairs that you can use for programming or manual lookup. For descriptions of the `ErrorData` values that an error code includes, see the troubleshooting page for the error code.

`ErrorMessage`

Description of the error with a link to more information in the Amazon EMR documentation.

> **Note**
>
> We don't recommend that you parse the text from `ErrorMessage` because this text is subject to change.

Error codes by category

- Bootstrap failure error codes in Amazon EMR
- Internal error codes for Amazon EMR
- Validation failure error codes in Amazon EMR

Bootstrap failure error codes in Amazon EMR

The following sections provide troubleshooting information for bootstrap failure error codes.

Topics

- BOOTSTRAP_FAILURE_PRIMARY_WITH_NON_ZERO_CODE

- BOOTSTRAP_FAILURE_BA_DOWNLOAD_FAILED_PRIMARY
- BOOTSTRAP_FAILURE_FILE_NOT_FOUND_PRIMARY
- BOOTSTRAP_FAILURE_INSUFFICIENT_DISK_SPACE_PRIMARY
- BOOTSTRAP_FAILURE_INSUFFICIENT_DISK_SPACE_WORKER
- BOOTSTRAP_FAILURE_HIVE_METASTORE_CONNECTION_ERROR_PRIMARY
- BOOTSTRAP_FAILURE_HIVE_METASTORE_CONNECTION_ERROR_WORKER

BOOTSTRAP_FAILURE_PRIMARY_WITH_NON_ZERO_CODE

Overview

When a cluster terminates with a `BOOTSTRAP_FAILURE_PRIMARY_WITH_NON_ZERO_CODE` error, a bootstrap action has failed in the primary instance. For more information about bootstrap actions, see Create bootstrap actions to install additional software with an Amazon EMR cluster.

Resolution

To resolve this error, review the details returned in the API error, modify your bootstrap action script, and create a new cluster with the updated bootstrap action.

To troubleshoot the failed EMR cluster, refer to the `ErrorDetail` information returned from the `DescribeCluster` and `ListClusters` APIs. For more information, see Error codes with ErrorDetail information in Amazon EMR. The `ErrorData` array within `ErrorDetail` returns the following information for this error code:

`primary-instance-id`

The ID of the primary instance where the bootstrap action failed.

`bootstrap-action`

The ordinal number for the bootstrap action that failed. A script with a `bootstrap-action` value of 1 is the first bootstrap action to run on the instance.

`return-code`

The return code for the bootstrap action that failed.

`amazon-s3-path`

The Amazon S3 location of the bootstrap action that failed.

`public-doc`

The public URL of the documentation for the error code.

Steps to complete

Perform the following steps to identify and fix the root cause of the bootstrap action error. Then launch a new cluster.

1. Review the bootstrap action log files in Amazon S3 to identify the root cause for the failure. To learn more on how to view Amazon EMR logs, see View Amazon EMR log files.
2. If you turned on cluster logs when you created the instance, refer to the `stdout` log for more information. You can find the `stdout` log for the bootstrap action in this Amazon S3 location:

   ```
   s3://amzn-s3-demo-bucket/logs/Your_Cluster_Id/node/Primary_Instance_Id/bootstrap-actions/Failed_Bootstrap_Action_Number/stdout.gz
   ```

   For more information on cluster logs, see Configure Amazon EMR cluster logging and debugging.
3. To determine the bootstrap action failure, review the exceptions in the `stdout` logs, and the `return-code` value in `ErrorData`.
4. Use your findings from the previous step to revise your bootstrap action so that it avoids exceptions or can gracefully handle exceptions when they occur.
5. Launch a new cluster with your updated bootstrap action.

BOOTSTRAP_FAILURE_BA_DOWNLOAD_FAILED_PRIMARY

Overview

A cluster terminates with the `BOOTSTRAP_FAILURE_BA_DOWNLOAD_FAILED_PRIMARY` error when the primary instance can’t download a bootstrap action script from the Amazon S3 location that you specify. Potential causes include the following:

- The bootstrap action script file isn’t in the specified Amazon S3 location.
- The service role for Amazon EC2 instances on the cluster (also called the *EC2 instance profile for Amazon EMR*) doesn't have permissions to access the Amazon S3 bucket where the bootstrap

action script resides. For more information about service roles, see Service role for cluster EC2 instances (EC2 instance profile).

For more information about bootstrap actions, see Create bootstrap actions to install additional software with an Amazon EMR cluster.

Resolution

To resolve this error, ensure that your primary instance has appropriate access to the bootstrap action script.

To troubleshoot the failed EMR cluster, refer to the `ErrorDetail` information returned from the `DescribeCluster` and `ListClusters` APIs. For more information, see Error codes with ErrorDetail information in Amazon EMR. The `ErrorData` array within `ErrorDetail` returns the following information for this error code:

`primary-instance-id`

The ID of the primary instance where the bootstrap action failed.

`bootstrap-action`

The ordinal number for the bootstrap action that failed. A script with a `bootstrap-action` value of `1` is the first bootstrap action to run on the instance.

`amazon-s3-path`

The Amazon S3 location of the bootstrap action that failed.

`public-doc`

The public URL of the documentation for the error code.

Steps to complete

Perform the following steps to identify and fix the root cause of the bootstrap action error. Then launch a new cluster.

Troubleshooting steps

1. Use the `amazon-s3-path` value from the `ErrorData` array to find the relevant bootstrap action script in Amazon S3.

2. If you turned on cluster logs when you created the instance, refer to the `stdout` log for more information. You can find the `stdout` log for the bootstrap action in this Amazon S3 location:

   ```
   s3://amzn-s3-demo-bucket/logs/Your_Cluster_Id/node/Primary_Instance_Id/bootstrap-actions/Failed_Bootstrap_Action_Number/stdout.gz
   ```

   For more information on cluster logs, see Configure Amazon EMR cluster logging and debugging.
3. To determine the bootstrap action failure, review the exceptions in the `stdout` logs, and the `return-code` value in `ErrorData`.
4. Use your findings from the previous step to revise your bootstrap action so that it avoids exceptions or can gracefully handle exceptions when they occur.
5. Launch a new cluster with your updated bootstrap action.

BOOTSTRAP_FAILURE_FILE_NOT_FOUND_PRIMARY

Overview

The `BOOTSTRAP_FAILURE_FILE_NOT_FOUND_PRIMARY` error indicates that the primary instance can't find the bootstrap action script that the instance just downloaded from the specified Amazon S3 bucket.

Resolution

To resolve this error, confirm that your primary instance has appropriate access to the bootstrap action script.

To troubleshoot the failed EMR cluster, refer to the `ErrorDetail` information returned from the `DescribeCluster` and `ListClusters` APIs. For more information, see Error codes with ErrorDetail information in Amazon EMR. The `ErrorData` array within `ErrorDetail` returns the following information for this error code:

`primary-instance-id`

The ID of the primary instance where the bootstrap action failed.

`bootstrap-action`

The ordinal number for the bootstrap action that failed. A script with a `bootstrap-action` value of 1 is the first bootstrap action to run on the instance.

`amazon-s3-path`

The Amazon S3 location of the bootstrap action that failed.

`public-doc`

The public URL of the documentation for the error code.

Steps to complete

Perform the following steps to identify and fix the root cause of the bootstrap action error. Then launch a new cluster.

1. To find the relevant bootstrap action script in Amazon S3, use the `amazon-s3-path` value from the `ErrorData` array.
2. Review the bootstrap action log files in Amazon S3 to identify the root cause for the failure. To learn more on how to view Amazon EMR logs, see View Amazon EMR log files.

   > **Note**
   >
   > If you didn't turn on logs for your cluster, you must create a new cluster with the same configurations and bootstrap actions. To ensure the cluster logs are turned on, see Configure Amazon EMR cluster logging and debugging.

3. Review the `stdout` log for your bootstrap actions and confirm that there are no custom processes that delete files in the `/emr/instance-controller/lib/bootstrap-actions` folder on your primary instances. You can find the `stdout` log for the bootstrap action in this Amazon S3 location:

   ```
   s3://amzn-s3-demo-bucket/logs/Your_Cluster_Id/node/Primary_Instance_Id/bootstrap-actions/Failed_Bootstrap_Action_Number/stdout.gz
   ```

4. Launch a new cluster with your updated bootstrap action.

BOOTSTRAP_FAILURE_INSUFFICIENT_DISK_SPACE_PRIMARY

Overview

The `BOOTSTRAP_FAILURE_INSUFFICIENT_DISK_SPACE_PRIMARY` error indicates that the primary instance does not have enough disk space when installing necessary software.

Resolution

To resolve this error, confirm that your primary instance has sufficient disk space on the root volume.

To troubleshoot the failed EMR cluster, refer to the `ErrorDetail` information returned from the `DescribeCluster` and `ListClusters` APIs. For more information, see Error codes with ErrorDetail information in Amazon EMR. The `ErrorData` array within `ErrorDetail` returns the following information for this error code:

`primary-instance-id`

The ID of the primary instance with insufficient disk space.

`public-doc`

The public URL of the documentation for the error code.

Steps to complete

1. Review the best practices for your cluster's EBS root device volume. See Customizing the Amazon EBS root device volume in the *Amazon EMR Management Guide*.
2. Launch a new cluster with a larger EBS root device volume size.

BOOTSTRAP_FAILURE_INSUFFICIENT_DISK_SPACE_WORKER

Overview

The `BOOTSTRAP_FAILURE_INSUFFICIENT_DISK_SPACE_WORKER` error indicates that one or more worker instances do not have enough disk space when installing necessary software.

Resolution

To resolve this error, confirm that your worker instances have sufficient disk space on the root volume.

To troubleshoot the failed EMR cluster, refer to the `ErrorDetail` information returned from the `DescribeCluster` and `ListClusters` APIs. For more information, see Error codes with ErrorDetail information in Amazon EMR. The `ErrorData` array within `ErrorDetail` returns the following information for this error code:

`worker-instance-ids`

The IDs of the worker instances with insufficient disk space.

`public-doc`

The public URL of the documentation for the error code.

Steps to complete

1. Review the best practices for your cluster's EBS root device volume. See Customizing the Amazon EBS root device volume in the *Amazon EMR Management Guide*.
2. Launch a new cluster with a larger EBS root device volume size.

BOOTSTRAP_FAILURE_HIVE_METASTORE_CONNECTION_ERROR_PRIMARY

Overview

The `BOOTSTRAP_FAILURE_HIVE_METASTORE_CONNECTION_ERROR_PRIMARY` error indicates that the primary instance is unable to establish a connection to the configured external Hive Metastore.

Resolution

To resolve this error, confirm that your external Hive Metastore is configured properly and the primary instance is allowed to connect to it.

To troubleshoot the failed EMR cluster, refer to the `ErrorDetail` information returned from the `DescribeCluster` and `ListClusters` APIs. For more information, see Error codes with ErrorDetail information in Amazon EMR. The `ErrorData` array within `ErrorDetail` returns the following information for this error code:

`primary-instance-id`

The ID of the primary instance unable to establish a connection to the configured external Hive Metastore.

`public-doc`

The public URL of the documentation for the error code.

Steps to complete

1. Review the best practices for for configuring an external metastore for Hive. See Configuring an external metastore for Hive.
2. Launch a new cluster with your updated cluster configuration.

BOOTSTRAP_FAILURE_HIVE_METASTORE_CONNECTION_ERROR_WORKER

Overview

The `BOOTSTRAP_FAILURE_HIVE_METASTORE_CONNECTION_ERROR_WORKER` error indicates that one or more worker instances are unable to establish a connection to the configured external Hive Metastore.

Resolution

To resolve this error, confirm that your external Hive Metastore is configured properly and the worker instances are allowed to connect to it.

To troubleshoot the failed EMR cluster, refer to the `ErrorDetail` information returned from the `DescribeCluster` and `ListClusters` APIs. For more information, see Error codes with ErrorDetail information in Amazon EMR. The `ErrorData` array within `ErrorDetail` returns the following information for this error code:

`worker-instance-ids`

The IDs of the worker instances unable to establish a connection to the configured external Hive Metastore.

`public-doc`

The public URL of the documentation for the error code.

Steps to complete

1. Review the best practices for for configuring an external metastore for Hive. See Configuring an external metastore for Hive.
2. Launch a new cluster with your updated cluster configuration.

Internal error codes for Amazon EMR

The following sections provide troubleshooting information for internal error codes, including codes for insufficient capacity or no capacity.

Topics

- INTERNAL_ERROR_EC2_INSUFFICIENT_CAPACITY_AZ
- INTERNAL_ERROR_SPOT_PRICE_INCREASE_PRIMARY
- INTERNAL_ERROR_SPOT_NO_CAPACITY_PRIMARY

INTERNAL_ERROR_EC2_INSUFFICIENT_CAPACITY_AZ

Overview

A cluster terminates with an `INTERNAL_ERROR_EC2_INSUFFICIENT_CAPACITY_AZ` error when the selected Availability Zone doesn't have enough capacity to fulfill your Amazon EC2 instance type request. The subnet that you select for a cluster determines the Availability Zone. For more information about subnets for Amazon EMR, see Configure networking in a VPC for Amazon EMR.

Resolution

To resolve this error, modify your instance type configurations and create a new cluster with your updated request.

To troubleshoot the failed EMR cluster, refer to the `ErrorDetail` information returned from the `DescribeCluster` and `ListClusters` APIs. For more information, see Error codes with ErrorDetail information in Amazon EMR. The `ErrorData` array within `ErrorDetail` returns the following information for this error code:

`instance-type`

The instance type that is out of capacity.

`availability-zone`

The Availability Zone that your subnet resolves to.

`public-doc`

The public URL of the documentation for the error code.

Steps to complete

Perform the following steps to identify and fix the root cause of the cluster configuration error:

- Review the best practices for cluster configuration. See Configuring Amazon EMR cluster instance types and best practices for Spot instances in the *Amazon EMR Management Guide*.
- Troubleshoot the launch issues and review your configuration. See Troubleshoot instance launch issues in the *Amazon EC2 User Guide*.
- Launch a new cluster with your updated cluster configuration.

INTERNAL_ERROR_SPOT_PRICE_INCREASE_PRIMARY

Overview

A cluster terminates with an `INTERNAL_ERROR_SPOT_PRICE_INCREASE_PRIMARY` error when Amazon EMR can't fulfill your Spot Instance request for the primary node because instances aren't available at or below your maximum Spot price. For more information, see Spot Instances in the *Amazon EC2 User Guide*.

Resolution

To resolve this error, specify instance types for your cluster that are within your price target, or increase your price limit for the same instance type.

To troubleshoot the failed EMR cluster, refer to the `ErrorDetail` information returned from the `DescribeCluster` and `ListClusters` APIs. For more information, see Error codes with ErrorDetail information in Amazon EMR. The `ErrorData` array within `ErrorDetail` returns the following information for this error code:

`primary-instance-id`

The ID for the primary instance of the cluster that failed.

`instance-type`

The instance type that is out of capacity.

`availability-zone`

The Availability Zone where your subnet resides.

`public-doc`

The public URL of the documentation for the error code.

Steps to complete

Perform the following steps to troubleshoot your cluster configuration strategy, and then launch a new cluster:

1. Review the best practices for Amazon EC2 Spot Instances and review your cluster configuration strategy. For more information, see Best practices for EC2 Spot in the *Amazon EC2 User Guide* and Configuring Amazon EMR cluster instance types and best practices for Spot instances.
2. Modify your instance type configurations or Availability Zone and create a new cluster with your updated request.
3. If the issue persists, use On-Demand capacity for your primary instance.

INTERNAL_ERROR_SPOT_NO_CAPACITY_PRIMARY

Overview

A cluster terminates with a `INTERNAL_ERROR_SPOT_NO_CAPACITY_PRIMARY` error when there isn't enough capacity to fulfill a Spot Instance request for your primary node. For more information, see Spot Instances in the *Amazon EC2 User Guide*.

Resolution

To resolve this error, specify instance types for your cluster that are within your price target, or increase your price limit for the same instance type.

To troubleshoot the failed EMR cluster, refer to the `ErrorDetail` information returned from the `DescribeCluster` and `ListClusters` APIs. For more information, see Error codes with ErrorDetail information in Amazon EMR. The `ErrorData` array within `ErrorDetail` returns the following information for this error code:

`primary-instance-id`

The ID for the primary instance of the cluster that failed.

`instance-type`

The instance type that is out of capacity.

`availability-zone`

The Availability Zone that your subnet resolves to.

`public-doc`

The public URL of the documentation for the error code.

Steps to complete

Perform the following steps to troubleshoot your cluster configuration strategy, and then launch a new cluster:

1. Review the best practices for Amazon EC2 Spot Instances and review your cluster configuration strategy. For more information, see Best practices for EC2 Spot in the *Amazon EC2 User Guide* and Configuring Amazon EMR cluster instance types and best practices for Spot instances.
2. Modify your instance type configurations and create a new cluster with your updated request.
3. If the issue persists, use On-Demand capacity for your primary instance.

Validation failure error codes in Amazon EMR

The following sections provide troubleshooting information for validation failure error codes.

Topics

- VALIDATION_ERROR_SUBNET_NOT_FROM_ONE_VPC
- VALIDATION_ERROR_SECURITY_GROUP_NOT_FROM_ONE_VPC
- VALIDATION_ERROR_INVALID_SSH_KEY_NAME
- VALIDATION_ERROR_INSTANCE_TYPE_NOT_SUPPORTED

VALIDATION_ERROR_SUBNET_NOT_FROM_ONE_VPC

Overview

When your cluster and the subnets that you reference for your cluster belong to different virtual private clouds (VPCs), the cluster terminates with a `VALIDATION_ERROR_SUBNET_NOT_FROM_ONE_VPC` error. You can launch clusters with Amazon EMR with the instance fleets configuration across subnets in a VPC. For more information about

instance fleets, see Planning and configuring instance fleets for your Amazon EMR cluster in the *Amazon EMR Management Guide*.

Resolution

To resolve this error, use subnets that belong to the same VPC as the cluster.

To troubleshoot the failed EMR cluster, refer to the `ErrorDetail` information returned from the `DescribeCluster` and `ListClusters` APIs. For more information, see Error codes with ErrorDetail information in Amazon EMR. The `ErrorData` array within `ErrorDetail` returns the following information for this error code:

`vpc`

For each subnet:VPC pair, the ID for the VPC that the subnet belongs to.

`subnet`

For each subnet:VPC pair, the ID for the subnet.

`public-doc`

The public URL of the documentation for the error code.

Steps to complete

Perform the following steps to identify and fix the error:

1. Review the subnet IDs that are listed in the `ErrorData` array and confirm that they belong to the VPC where you want to launch the EMR cluster.
2. Modify your subnet configurations. You can use one of the following methods to find all available public and private subnets in a VPC.
   - Navigate to the Amazon VPC Console. Choose **Subnets** and list all of the subnets that reside within the AWS Region for your cluster. To find only public or private subnets, apply the **Auto-assign public IPv4 address** filter. To find and select subnets in the VPC that your cluster uses, use the **Filter by VPC** option. For more information on how to create subnets, see Create a subnet in the *Amazon Virtual Private Cloud User Guide*.
   - Use the AWS CLI to find all available public and private subnets in the VPC that your cluster uses. For more information, see the describe-subnets API. To create new subnets in a VPC, see the create-subnet API.

3. Launch a new cluster with subnets from the same VPC as the cluster.

VALIDATION_ERROR_SECURITY_GROUP_NOT_FROM_ONE_VPC

Overview

When your cluster and the security groups that you assign to your cluster belong to different virtual private clouds (VPCs), the cluster terminates with a `VALIDATION_ERROR_SECURITY_GROUP_NOT_FROM_ONE_VPC` error. For more information about security groups, see Specifying Amazon EMR-managed and additional security groups and Control network traffic with security groups for your Amazon EMR cluster.

Resolution

To resolve this error, use security groups that belong to the same VPC as the cluster.

To troubleshoot the failed EMR cluster, refer to the `ErrorDetail` information returned from the `DescribeCluster` and `ListClusters` APIs. For more information, see Error codes with ErrorDetail information in Amazon EMR. The `ErrorData` array within `ErrorDetail` returns the following information for this error code:

`vpc`

For each security-group:VPC pair, the ID for the VPC that the security group belongs to.

`security-group`

For each security-group:VPC pair, the ID for the security group.

`public-doc`

The public URL of the documentation for the error code.

Steps to complete

Perform the following steps to identify and fix the error:

1. Review the security group IDs that are listed in the `ErrorData` array and confirm that they belong to the VPC where you want to launch the EMR cluster.
2. Navigate to the Amazon VPC Console. Choose **Security groups** to list all of the security groups within the Region that you select. Find the security groups from the same VPC as your cluster, and then modify your security group configuration.

3. Launch a new cluster with security groups from the same VPC as the cluster.

VALIDATION_ERROR_INVALID_SSH_KEY_NAME

Overview

A cluster terminates with a `VALIDATION_ERROR_INVALID_SSH_KEY_NAME` error when you use an Amazon EC2 key pair that isn't valid to SSH into the primary instance. The key pair name might be incorrect, or the key pair might not exist in the requested AWS Region. For more information about key pairs, see Amazon EC2 key pairs and Linux instances in the *Amazon EC2 User Guide*.

Resolution

To resolve this error, create a new cluster with a valid SSH key pair name.

To troubleshoot the failed EMR cluster, refer to the `ErrorDetail` information returned from the `DescribeCluster` and `ListClusters` APIs. For more information, see Error codes with ErrorDetail information in Amazon EMR. The `ErrorData` array within `ErrorDetail` returns the following information for this error code:

`ssh-key`

The SSH key pair name that you provided when you created the cluster.

`public-doc`

The public URL of the documentation for the error code.

Steps to complete

Perform the following steps to identify and fix the error:

1. Check your *`keypair`*.pem file and confirm that it matches the name of the SSH key that you see in the Amazon EMR console.
2. Navigate to the Amazon EC2 console. Verify that the SSH key name that you used is available in the AWS Region that your cluster uses. You can find your AWS Region next to your account ID at the top of the AWS Management Console.
3. Launch a new cluster with a valid SSH key name.

VALIDATION_ERROR_INSTANCE_TYPE_NOT_SUPPORTED

Overview

A cluster terminates with a `VALIDATION_ERROR_INSTANCE_TYPE_NOT_SUPPORTED` error when the AWS Region and Availability Zones for your cluster don't support the specified instance type for one or more instance groups. Amazon EMR might support an instance type in one Availability Zone within a Region but not another. The subnet that you select for a cluster determines the Availability Zone within the Region. For a list of instance types and Regions that Amazon EMR supports, see Supported instance types with Amazon EMR.

Resolution

To resolve this error, specify instance types for your cluster that Amazon EMR supports in the Region and Availability Zone where you request the cluster.

To troubleshoot the failed EMR cluster, refer to the `ErrorDetail` information returned from the `DescribeCluster` and `ListClusters` APIs. For more information, see Error codes with ErrorDetail information in Amazon EMR. The `ErrorData` array within `ErrorDetail` returns the following information for this error code:

`instance-types`

The list of unsupported instance types.

`availability-zones`

The list of Availability Zones that your subnet resolves to.

`public-doc`

The public URL of the documentation for the error code.

Steps to complete

Perform the following steps to identify and fix the error:

1. Use the AWS CLI to retrieve the available instance types in an Availability Zone. To do this, you can use the `ec2 describe-instance-type-offerings` command to filter available instance types by location (AWS Region or Availability Zone). For example, the following command returns the instance types that are offered in the specified AZ, *`us-east-2a`*.

```
aws ec2 describe-instance-type-offerings --location-type "availability-
zone" --filters Name=location,Values=us-east-2a --region us-east-2 --query
 "InstanceTypeOfferings[*].[InstanceType]" --output text | sort
```

To learn more about how to discover available instance types, see Find an Amazon EC2 instance type.

2. After you determine the instance types that are available in the same Region and Availability Zone as the cluster, choose one of the following resolutions to continue:

   a. Create a new cluster, and choose a subnet for the cluster that is in an Availability Zone where the instance type that you selected is available and supported by Amazon EMR.

   b. Create a new cluster in the same Region and Amazon EC2 subnet as the cluster that failed, but with an instance type that is supported in that location by Amazon EMR.

For a list of instance types and Regions that Amazon EMR supports, see Supported instance types with Amazon EMR. To compare the capabilities of the instance types, see Amazon EC2 instance types.

Resource errors during Amazon EMR cluster operations

The following errors are commonly caused by constrained resources on the cluster. The guidance describes each error and provides troubleshooting help.

Topics

- Amazon EMR cluster terminates with NO_SLAVE_LEFT and core nodes FAILED_BY_MASTER
- Amazon EMR cluster error: Cannot replicate block, only managed to replicate to zero nodes.
- Amazon EMR cluster error: EC2 QUOTA EXCEEDED
- Amazon EMR cluster error: Too many fetch-failures
- Amazon EMR cluster error: File could only be replicated to 0 nodes instead of 1
- Amazon EMR cluster error: Deny-listed nodes
- Throttling errors with an Amazon EMR cluster
- Amazon EMR cluster error: Instance type not supported
- Amazon EMR cluster error: EC2 is out of capacity
- Amazon EMR cluster error: HDFS replication factor error

- Amazon EMR cluster error: HDFS insufficient space error

Amazon EMR 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. 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 Use Amazon EMR cluster scaling to adjust for changing workloads.

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 to protect your Amazon EMR clusters from accidental shut down.

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 Monitoring Amazon EMR metrics with CloudWatch.

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`	0	The minimum space that must be available on a disk for it to be used. This

Property	Default value	Description
`.min-free-space-per-disk-mb`		applies to `yarn-node manager.local-dirs` and `yarn.nodemanager.log-dirs` .

Amazon EMR cluster error: 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*.

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.

Amazon EMR cluster error: 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.
- 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.

Amazon EMR cluster error: 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 with an Amazon EMR cluster.
- 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 Amazon EMR cluster hardware and networking.
- 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.

Amazon EMR cluster error: 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 resize a running Amazon EMR cluster and Monitoring Amazon EMR metrics with CloudWatch.

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 Amazon EMR log files.

Amazon EMR cluster error: 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 with an Amazon EMR cluster

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 resize a running Amazon EMR cluster.
- 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.

- Instance fleets in a cluster add instances to meet an increased target capacity. For more information, see Planning and configuring instance fleets for your Amazon EMR cluster.

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.
- 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 in Amazon EMR.

Amazon EMR cluster error: 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 with Amazon EMR. 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.

Mitigate instance fleet cluster launch failures because of unsupported primary instance types in Amazon EMR

Primary nodes are essential in Amazon EMR clusters. An EMR cluster launch might fail with an `instance type not supported` error where Amazon EMR attempts to launch the cluster in an Availability Zone where if the primary instance type isn't supported. Enhanced Availability Zone selection for instance fleet clusters in Amazon EMR automatically filters out unsupported AZs for the primary instance types you specified in the cluster configuration. This means Amazon EMR won't choose an Availability Zone where the configured primary instance types are not supported, which prevents cluster launch failures because of unsupported instance types.

To enable this improvement, add the required permission to the service role or policy for your cluster. The latest version of `AmazonEMRServicePolicy_v2` includes this permission, so if you use that policy, the improvement is already available. If you use a custom service role or policy, add the permission `ec2:DescribeInstanceTypeOfferings` when launching your cluster.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:DescribeInstanceTypeOfferings"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ],
      "Sid": "AllowEC2Describeinstancetypeofferings"
    }
  ]
}
```

Amazon EMR cluster error: 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 with Amazon EMR. 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.

Amazon EMR cluster error: HDFS replication factor error

When you remove a core node from a core instance group or instance fleet, Amazon EMR might run into an HDFS replication error. This error happens when you remove core nodes and the number core nodes falls below the configured dfs.replication factor for the Hadoop Distributed File System (HDFS). As such, Amazon EMR can't safely perform the operation. To determine the default value of the `dfs.replication` configuration, HDFS configuration.

Possible causes

See the following for the possible causes of HDFS replication factor error:

- If you manually resize a core instance group or instance fleet below the configured `dfs.replication` factor.
- Your policies for managed scaling or autoscaling might allow for scaling to reduce the number of core nodes below the threshold of `dfs.replication`.
- This error can also occur if Amazon EMR tries to replace an unhealthy core node when a cluster has the minimal number of core nodes defined by `dfs.replication`.

Solutions and best practices

See the following for solutions and best practices:

- When you manually resize an Amazon EMR cluster, don't scale down below the `dfs.replication` as Amazon EMR can't safely complete the resize.
- When you use managed scaling or autoscaling, make sure that the minimum capacity of your cluster isn't lower than the `dfs.replication` factor.
- The number of core instances should be at least `dfs.replication` plus one. This makes sure that Amazon EMR can successfully replace an unhealthy core node if you enabled unhealthy core replacement.

> **Important**
>
> Failure of a single core node can lead to HDFS data loss if you set `dfs.replication` to 1. If your cluster has HDFS storage, we recommend that you configure the cluster with at least four core nodes for production workloads to avoid data loss and also set the `dfs.replication` factor to at least 2.

Amazon EMR cluster error: HDFS insufficient space error

An Hadoop Distributed File System (HDFS) insufficient space error can occur if you attempt to remove a core node, but Amazon EMR can't safely complete the operation because of insuficicent space left in the HDFS. Before Amazon EMR removes a core node, all HDFS data on the node must be transferred to other core nodes to ensure data redundancy. However, if there isn't enough space on the other core nodes for replication, Amazon EMR can't gracefully decommission the node.

Possible causes

See the following for a list of possible causes of HDFS insufficient space error:

- If you manually scale down a core instance group or instance fleet when there isn't enough HDFS space on the remaining nodes for data replication prior to the scale down.
- Managed scaling or autoscaling scales down a core instance group or instance fleet when there isn't enough HDFS space for data replication.
- Amazon EMR attempts to replace an unhealthy core node but is unable to safely replace the node due to insufficient HDFS space.

Solutions and best practices

See the following for solutions and best practices:

- Scale up the number of core nodes in your Amazon EMR cluster. If you use managed scaling or autoscaling, increase the minimum capacity of your core nodes.
- Use larger EBS volumes for your core nodes when you create your EMR cluster.
- Delete unneeded HDFS data in your EMR cluster. We recommend that you set up CloudWatch alarms to monitor the `HDFSUtilization` metric in your cluster to know if your EMR cluster is low on space.

Cluster input and output errors during Amazon EMR operations

The following errors are common in cluster input and output operations. Use the guidance in this topic to troubleshoot and check your configuration.

Topics

- Does your path to Amazon Simple Storage Service (Amazon S3) have at least three slashes?

- Are you trying to recursively traverse input directories?
- Does your output directory already exist?
- Are you trying to specify a resource using an HTTP URL?
- Are you referencing an Amazon S3 bucket using an invalid name format?
- Are you experiencing trouble loading data to or from Amazon S3?

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://amzn-s3-demo-bucket1", you should use "s3n://amzn-s3-demo-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 "amzn-s3-demo-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?.

For additional information, see Amazon S3 error best practices in the *Amazon Simple Storage Service User Guide*.

Permissions errors during Amazon EMR cluster operations

The following errors are common when using permissions or credentials.

Topics

- Are you passing the correct credentials into SSH?
- If you are using IAM, do you have the proper Amazon EC2 policies set?

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 Amazon EMR cluster primary node using SSH.

If you are using IAM, do you have the proper Amazon EC2 policies set?

Because Amazon EMR uses EC2 instances as nodes, a 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 a user's behalf. If you do not have the required permissions set, Amazon EMR returns the error: **"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.

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?
- Did you encounter a syntax error in the Hive script?
- Did a job fail when running interactively?
- Are you having trouble loading data to or from Amazon S3 into Hive?

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 Amazon EMR log files.

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 Amazon EMR log files.

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?. 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 during Amazon EMR cluster operations

The following errors are common to VPC configuration in Amazon EMR.

Topics

- Invalid subnet configuration
- Missing DHCP Options Set
- Permissions errors
- Errors that result in START_FAILED
- Cluster Terminated with errors and NameNode fails to start

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://amzn-s3-demo-bucket/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.

Streaming Amazon EMR 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?
- Is your script timing out?
- Are you passing in invalid streaming arguments?
- Did your script exit with an error?

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 Amazon EMR log files.

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 Amazon EMR log files.

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 Amazon EMR log files.

Amazon EMR: Custom JAR cluster errors

The following errors are common to custom JAR clusters.

Topics

- Is your JAR throwing an exception before creating a job?
- Is your JAR throwing an error inside a map task?

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 Amazon EMR log files.

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 Amazon EMR log files.

Amazon EMR 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.
- 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 during Amazon EMR cluster operations.

If these steps do not solve the problem, continue with the steps for troubleshooting common Amazon EMR errors. For more information, see Collections of common errors in Amazon EMR.

Find a missing cluster

If your cluster is missing from the console list or `ListClusters` API, check the following:

- Confirm that the cluster age from time of completion is less than two months. Amazon EMR preserves metadata information for completed clusters for two months at no charge. You can't delete completed clusters from the console — instead, Amazon EMR purges completed clusters automatically after two months.
- Confirm that you have role permissions to view the cluster.
- Confirm that you are viewing the same AWS Region where the cluster resides.

Troubleshoot an Amazon EMR cluster that has failed with an error code

This section walks you through the process of troubleshooting a cluster that has failed. This means that the cluster terminated with an error code.

> **Note**
>
> When an EMR cluster terminates with an error, the `DescribeCluster` and `ListClusters` APIs return an error code and an error message. For some cluster errors, the `ErrorDetail` data array can also help you troubleshoot the failure. For more information, see Error codes with ErrorDetail information in Amazon EMR.

If your cluster runs but takes a long time to return results, see Troubleshoot a slow Amazon EMR cluster.

Topics

- Step 1: Gather data about the issue with the Amazon EMR cluster
- Step 2: Check the environment
- Step 3: Look at the last state change
- Step 4: Examine the Amazon EMR log files
- Step 5: Test the Amazon EMR cluster step by step

Step 1: Gather data about the issue with the Amazon EMR cluster

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 Amazon EMR cluster status and details.

- Identifier of the cluster. (Also called a job flow identifier.)
- AWS 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
- Check usage limits
- Check the release version
- Check the Amazon VPC subnet configuration

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, 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 a 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 in a VPC for Amazon EMR. 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 Amazon EMR cluster status and details.

Step 4: Examine the Amazon EMR 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 Amazon EMR log files.

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 Amazon EMR 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 an Amazon EMR cluster to continue or terminate after step execution and Using termination protection to protect your Amazon EMR clusters from accidental shut down.
2. Submit a step to the cluster. For more information, see Submit work to an Amazon EMR cluster.

3. When the step completes processing, check for errors in the step log files. For more information, see Step 4: Examine the Amazon EMR log files. 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 to protect your Amazon EMR clusters from accidental shut down.

Troubleshoot a slow Amazon EMR 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 an Amazon EMR cluster that has failed with an error code

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 Amazon EMR cluster hardware and networking.

The following topics walk you through the process of identifying alternative causes of a slow cluster.

Topics

- Step 1: Gather data about the issue with the Amazon EMR cluster
- Step 2: Check the EMR cluster environment
- Step 3: Examine the log files for the Amazon EMR cluster
- Step 4: Check Amazon EMR cluster and instance health
- Step 5: Check for suspended groups
- Step 6: Review configuration settings for the Amazon EMR cluster

- Step 7: Examine input data for the Amazon EMR cluster

Step 1: Gather data about the issue with the Amazon EMR cluster

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 Amazon EMR cluster status and details.

- Identifier of the cluster. (Also called a job flow identifier.)
- AWS 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 EMR cluster environment

Check your environment to see if there are service outages or you have exceeded an AWS service limit.

Topics

- Check for service outages
- Check usage limits
- Check the Amazon VPC subnet configuration
- Restart the cluster

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, 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 a 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 in a VPC for Amazon EMR. 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 for the Amazon EMR cluster

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 Amazon EMR log files.

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 Amazon EMR 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 resize a running Amazon EMR cluster.
- **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 resize a running Amazon EMR cluster.

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 Monitoring Amazon EMR metrics with CloudWatch.

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 Amazon EMR application history.

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.

- 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.

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.

Step 6: Review configuration settings for the Amazon EMR cluster

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 with an Amazon EMR cluster. 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/.

Warning

1. Setting `dfs.replication` to 1 on clusters with fewer than four nodes can lead to HDFS data loss if a single node goes down. We recommend you use a cluster with at least four core nodes for production workloads.
2. Amazon EMR will not allow clusters to scale core nodes below `dfs.replication`. For example, if `dfs.replication = 2`, the minimum number of core nodes is 2.
3. When you use Managed Scaling, Auto-scaling, or choose to manually resize your cluster, we recommend that you to set `dfs.replication` to 2 or higher.

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

Configuration setting	Description
	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.speculativ e.execution	Determines whether map task attempts of the same task may be launched in parallel.
mapred.reduce.tasks.specula tive.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.com pleted.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 for the Amazon EMR cluster

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.

Troubleshoot common issues when using Amazon EMR with AWS Lake Formation

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.

Querying cross-account data shared with Lake Formation

When you use Amazon EMR to access data shared with you from another account, some Spark libraries will attempt to call `Glue:GetUserDefinedFunctions` API operation. Since versions 1 and 2 of the AWS RAM managed permissions does not support this action, you receive the following error message:

```
"ERROR: User: arn:aws:sts::012345678901:assumed-role/my-
spark-role/i-06ab8c2b59299508a is not authorized to perform:
glue:GetUserDefinedFunctions on resource: arn:exampleCatalogResource
because no resource-based policy allows the glue:GetUserDefinedFunctions
action"
```

To resolve this error, the data lake administrator who created the resource share must update the AWS RAM managed permissions attached to the resource share. Version 3 of the AWS RAM managed permissions allows principals to perform the `glue:GetUserDefinedFunctions` action.

If you create a new resource share, Lake Formation applies the latest version of the AWS RAM managed permission by default, and no action is required by you. To enable cross-account data access for existing resource shares, you need to update the AWS RAM managed permissions to version 3.

You can view the AWS RAM permissions assigned to resources shared with you in AWS RAM. The following permissions are included in version 3:

```
Databases
  AWSRAMPermissionGlueDatabaseReadWriteForCatalog
  AWSRAMPermissionGlueDatabaseReadWrite

Tables
  AWSRAMPermissionGlueTableReadWriteForCatalog
  AWSRAMPermissionGlueTableReadWriteForDatabase

AllTables
  AWSRAMPermissionGlueAllTablesReadWriteForCatalog
  AWSRAMPermissionGlueAllTablesReadWriteForDatabase
```

To update AWS RAM managed permissions version of existing resource shares

You (data lake administrator) can either update AWS RAM managed permissions to a newer version by following instructions in the *AWS RAM User Guide* or you can revoke all existing permissions for the resource type and regrant them. If you revoke permissions, AWS RAM deletes the AWS RAM resource share associated with the resource type. When you regrant permissions, AWS RAM creates new resource shares attaching the latest version of AWS RAM managed permissions.

Inserting into, creating, and altering tables

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 Amazon EMR clusters

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.

Topics

- End-to-end Amazon EMR Java source code sample
- Common concepts for Amazon EMR API calls
- Use SDKs to call Amazon EMR APIs
- Manage Amazon EMR Service Quotas

> **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.

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 Amazon EMR API calls

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.

Topics

- Endpoints for Amazon EMR
- Specifying cluster parameters in Amazon EMR
- Availability Zones in Amazon EMR
- How to use additional files and libraries in Amazon EMR clusters

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.

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

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
- How to manage Amazon EMR Service Quotas
- When to set up EMR events in CloudWatch

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 AWS CLI. To learn more about viewing quotas and requesting increases, see AWS service quotas in the *Amazon Web Services General Reference*.

> **Important**
>
> Service quotas can be enforced independent of each other and at times their effects can overlap. Some are scoped to specific APIs, and other quotas enforce limits at the account level. Specifically, the following quotas are applied at the account level:
>
> - *The maximum rate at which your bucket replenishes for all EMR operations*
> - *The maximum number of API requests that you can make per second*
>
> Note that if you successfully request a specific quota increase and you continue to see throttling, for example for API requests, it's possible that one of these account-level quotas

are continuing to limit API call rates. For troubleshooting, you can use the Service quotas console, which is available at https://console.aws.amazon.com/servicequotas/home, to check quota limits and request increases. If you make requests for increases and continue to see throttling, contact support. You can also find default limits and descriptions for EMR Service quotas in the **AWS General Reference** *Reference guide*.

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.

When to set up EMR events in CloudWatch

For some polling APIs, such as DescribeCluster, DescribeStep, and ListClusters, setting up a CloudWatch 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 Glossary Reference*.

Document history

The following table describes notable changes to the Amazon EMR Management Guide since the last release of Amazon EMR. For more information about specific released versions of Amazon EMR, see About Amazon EMR Releases.

Change	Description	Release date
Managed policy update	Amazon EMR updates to AWS managed policies – Amazon EMR updates to AWS managed policies – Additional permissions for AmazonEMRServicePolicy_v2.	March 4, 2025
Initial Release	Initial release of the Amazon EMR Management Guide for versions 4.x and later of Amazon EMR. For more information about previous versions of Amazon EMR, see Amazon EMR Developer Guide.	July 24, 2015